

RASLAN 2022
Recent Advances in Slavonic
Natural Language Processing

A. Horák, P. Rychlý, A. Rambousek (eds.)

RASLAN 2022

**Recent Advances in Slavonic Natural
Language Processing**

**Sixteenth Workshop on Recent Advances
in Slavonic Natural Language Processing,
RASLAN 2022**

**Karlova Studánka, Czech Republic,
December 9–11, 2022
Proceedings**

**Tribun EU
2022**

Proceedings Editors

Aleš Horák
Faculty of Informatics, Masaryk University
Department of Information Technologies
Botanická 68a
CZ-602 00 Brno, Czech Republic
Email: hales@fi.muni.cz

Pavel Rychlý
Faculty of Informatics, Masaryk University
Department of Information Technologies
Botanická 68a
CZ-602 00 Brno, Czech Republic
Email: pary@fi.muni.cz

Adam Rambousek
Faculty of Informatics, Masaryk University
Department of Information Technologies
Botanická 68a
CZ-602 00 Brno, Czech Republic
Email: rambousek@fi.muni.cz

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the Czech Copyright Law, in its current version, and permission for use must always be obtained from Tribun EU. Violations are liable for prosecution under the Czech Copyright Law.

Editors © Aleš Horák, 2022; Pavel Rychlý, 2022; Adam Rambousek, 2022

Typography © Adam Rambousek, 2022

Cover © Petr Sojka, 2010

This edition © Tribun EU, Brno, 2022

ISBN 978-80-263-1752-4

ISSN 2336-4289

Preface

This volume contains the Proceedings of the Sixteenth Workshop on Recent Advances in Slavonic Natural Language Processing (RASLAN 2022), organized by the NLP Consulting, s.r.o. and held on December 9th–11th 2022 in Karlova Studánka, Sporthotel Kurzovní, Jeseníky, Czech Republic.

The RASLAN Workshop is an event dedicated to the exchange of information between research teams working on the projects of computer processing of Slavonic languages and related areas going on in the NLP Centre at the Faculty of Informatics, Masaryk University, Brno. RASLAN is focused on theoretical as well as technical aspects of the project work, on presentations of verified methods together with descriptions of development trends. The workshop also serves as a place for discussion about new ideas. The intention is to have it as a forum for presentation and discussion of the latest developments in the field of language engineering, especially for undergraduates and postgraduates affiliated to the NLP Centre at FI MU.

Topics of the Workshop cover a wide range of subfields from the area of artificial intelligence and natural language processing including (but not limited to):

- * text corpora and tagging
- * syntactic parsing
- * sense disambiguation
- * machine translation, computer lexicography
- * semantic networks and ontologies
- * semantic web
- * knowledge representation
- * logical analysis of natural language
- * applied systems and software for NLP

RASLAN 2022 offers a rich program of presentations, short talks, technical papers and mainly discussions. A total of 23 papers were accepted, contributed altogether by 44 authors. Our thanks go to the Program Committee members and we would also like to express our appreciation to all the members of the Organizing Committee for their tireless efforts in organizing the Workshop and ensuring its smooth running. In particular, we would like to mention the work of Aleš Horák, Pavel Rychlý and Jitka Nováčková. The T_EXpertise of Adam Rambousek (based on L^AT_EX macros prepared by Petr Sojka) resulted in the extremely speedy and efficient production of the volume which you are now holding in your hands. Last but not least, the cooperation of Tribun EU as a publisher and printer of these proceedings is gratefully acknowledged.

Brno, December 2022

Aleš Horák

Table of Contents

I NLP Applications

Parallel, or Comparable? That Is the Question	3
<i>Michaela Denisová</i>	
Automatic Identification of Speakers and Parties in Steno Protocols of the Czech Parliament	15
<i>Ota Mikušek</i>	
Keyness Analysis and Its Representation in Russian Academic Papers on Computational Linguistics: Evaluation of Algorithms	25
<i>Maria Khokhlova and Mikhail Koryshev</i>	
Information Extraction from Business Documents: A Case Study	35
<i>Martin Geletka, Mikuláš Bankovič, Dávid Meluš, Šárka Ščavnická, Michal Štefánek, and Petr Sojka</i>	
Keyword Extraction for Automatic Evaluation of Machine Translation	47
<i>Lívia Kelebercová and František Forgáč</i>	
Using NVH as a Backbone Format in the Lexonomy Dictionary Editor . . .	55
<i>Miloš Jakubíček, Vojtěch Kovář, Michal Měchura, and Adam Rambousek</i>	

II Evaluation Methods

Are Dictionary Definitions of Verbs in Corpora?	65
<i>Marie Stará</i>	
Evaluation of Various Approaches to Compute BLEU Metrics	71
<i>Lucia Benková, and Lubomír Benko</i>	
Compressed FastText Models for Czech Tagger	79
<i>Zuzana Nevěřilová</i>	
Blooming Onion: Efficient Deduplication through Approximate Membership Testing	91
<i>Ondřej Herman</i>	
CompAn – A Tool for Quantitative Comparison of Corpus Annotation . . .	97
<i>Vlasta Ohlidalová</i>	

III Text Corpora

Piötòst Ché Niènt, Mèi Piötòst - A Manually Revised Lombard-Italian Parallel Corpus	105
<i>Edoardo Signoroni</i>	
Aranea Go Middle East: Persicum	113
<i>Vladimír Benko</i>	
Pipeline Effectiveness in the Sketch Engine	123
<i>Matúš Kostka</i>	
Constructing Datasets from Dialogue Data	131
<i>Ondřej Sotolář, Jaromír Plhák, Michaela Lebedíková, Michał Tkaczyk, and David Šmahel</i>	
Semi-Manual Annotation of Topics and Genres in Web Corpora, The Cheap and Fast Way	141
<i>Vít Suchomel, Jan Kraus</i>	
Utok: The Fast Rule-based Tokenizer	149
<i>Pavel Rychlý and Samuel Špalek</i>	

IV Semantics and Language Modelling

When Tesseract Meets PERO: Open-Source Optical Character Recognition of Medieval Texts	157
<i>Vít Novotný and Aleš Horák</i>	
Medical Knowledge Resources for Text-Mining of Health Records in Czech, Polish, and Slovak	163
<i>Krištof Anetta</i>	
Secondary Prepositions with Causal Meaning in Russian	173
<i>Victor Zakharov, Kirill Boyarsky, Eugeny Kanevsky, and Anastasia Kozlova</i>	
Towards General Document Understanding through Question Answering	183
<i>Šárka Ščavnická, Michal Štefánik, Marek Kadlčík, Martin Geletka, and Petr Sojka</i>	
Manipulative Style Recognition of Czech News Texts using Stylometric Text Analysis	191
<i>Radoslav Sabol and Aleš Horák</i>	

The Influence of a Machine Translation System on Sentiment Levels	201
<i>Jaroslav Reichel, and Lubomír Benko</i>	
Subject Index	209
Author Index	211

Part I

NLP Applications

Parallel, or Comparable? That Is the Question

The Comparison of Parallel and Comparable Data-based Methods for Bilingual Lexicon Induction

Michaela Denisová

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
449884@mail.muni.cz

Abstract. Extracting translation equivalents from parallel data has been considered the main and most efficient method in the lexicography field. However, parallel data are not always available or sufficient, especially for rare and low-resource language pairs. Translation equivalents obtained from comparable data offer a solution for this problem. This paper compares the performance of some methods that utilize either parallel or comparable data and demonstrates the results on the Estonian-Slovak language combination. We show that comparable data usage aspires to be a viable alternative for low-resource languages or rare language pairs, and we propose a new equation for more effective translation equivalents' induction.

Keywords: Parallel data, Comparable data, Translation equivalents' extraction, Estonian, Slovak

1 Introduction

Over many years, inducing translation equivalents through parallel data has been a preferred method among lexicographers. Parallel data often means parallel corpora or, in some cases, bilingual dictionaries. Current lexicographic tools utilizing either of those provide an effective and reliable method for obtaining translation equivalents as they include a lot of context information.[5]

However, rare language pairs or low-resource languages often lack parallel data, which could mirror in the quality and amount of the resulting translation equivalents. An alternative offer quickly developing modern approaches from the NLP field that claim using only comparable data for this task as sufficient. Among these methods are cross-lingual embedding models requiring monolingual word embeddings and only a few or no supervision signals at all. These models are evaluated on various tasks, such as cross-lingual named entity recognition, information retrieval, etc. This paper focuses mainly on the bilingual lexicon induction task, i.e., the BLI task.

The drawback of the comparable data-based methods is that they do not involve any context information as they are one-to-one or one-to-many alignments. Therefore, in contrast to the parallel data-based methods, they exclude

any phrases or multi-word expressions that are valid parts of bilingual dictionaries.

This paper aims to compare the quality of the resulting translation equivalents obtained by chosen methods that utilize either comparable or parallel data. We show the results on a rare language combination, namely Estonian-Slovak. We induce a certain amount of translation equivalents with each method and evaluate them manually.

Our motivation is to explore whether recent trends favouring comparable data can compete with standard, widely used parallel data.

In this case, the bilingual dictionary-based parallel data method is represented by a pivot Estonian-Slovak dictionary obtained from English-Estonian and English-Slovak dictionaries [8]. Regarding parallel corpus, we manually evaluated the Estonian-Slovak dictionary extracted from the Estonian-Slovak parallel corpus EUR-Lex from SketchEngine.[6]

For the comparable data, we picked three currently most cited cross-lingual embedding models that are often used as benchmarks, MUSE [7,14], VecMAP [1,2,4,3], and FastText for bilingual alignment [11]. Moreover, we experiment with different levels of supervision.

This paper is structured as follows. In Section 2, we introduce the chosen methods for our comparison and divide them into comparable and parallel-data-based. In Section 3, we explain the data we used for the evaluation in further detail. In Section 4, we present our results and provide a thorough comparison of the evaluated models. In Section 5, we offer concluding remarks and outline new ideas for future work.

2 Related Work

This section provides insight into methods used in this paper for obtaining translation equivalents from either parallel or comparable data.

2.1 Comparable Data

One of the solutions for extracting translation equivalents using comparable data provides cross-lingual embedding models. Cross-lingual embedding models have recently become a popular research topic as they can connect meanings across languages. The monolingual word embeddings of two or multiple languages are projected into a shared joint space where words with similar meanings obtain similar vectors. Afterwards, translation equivalents' candidates are extracted by computing cosine similarity.[15]

Frequently, the methods use various levels of supervision; they can be strongly supervised, semi-supervised or unsupervised. Supervision signals are represented by word-to-word dataset and vary from 5,000 words or can be comprised of similar strings and numerals.

These models offer a good solution for low-resource languages or rare language pairs because they do not require extensive parallel data. On the other

hand, they are still behind their parallel data-based counterparts concerning multi-word expressions or phrases as they do not include any context information.

In the following subsections, we describe three approaches we choose for our experiment. The reason behind this is that these approaches are stated in many papers as benchmarks and considered state-of-the-art models among cross-lingual models.

MUSE is a framework that combines domain-adversarial settings with applying the iterative Procrustes algorithm. The model can be trained in a supervised or unsupervised manner. Furthermore, it provides an option to rely only on identical strings. Code and pre-trained aligned word embeddings are available in an open-source GitHub library.¹

VECMap is a robust framework that consists of multiple steps, including iterative refinement and bootstrapping techniques. Similarly to MUSE, it has multiple types of training, such as supervised, semi-supervised, training relying on identical strings and unsupervised training. Moreover, the library and code are available on GitHub.²

FastText utilizes orthogonal mapping and modified CSLS retrieving method.[11] Script is available on the GitHub repository³ and pre-trained aligned word embeddings are available on FastText official website.⁴

2.2 Parallel Data

As mentioned above, this paper recognizes two types of parallel data: parallel corpora and bilingual dictionaries.

Parallel corpora usage has been the preferred approach among lexicographers as it produces high-quality dictionaries. The crucial argument is that parallel corpora contain rich context information, lowering human intuition in building a bilingual dictionary.[5]

The downside of parallel corpora is that it does not offer enough data for small languages or uncommon language pairs. The parallel corpora-driven bilingual dictionaries for such languages do not cover a sufficient amount of information for their users.

In this experiment, we manually evaluate Estonian-Slovak translation equivalents extracted from the Estonian-Slovak parallel corpus EUR-Lex with around 300,000,000 tokens. EUR-Lex is a multilingual corpus composed of texts from

¹ <https://github.com/facebookresearch/MUSE>

² <https://github.com/artetxem/vecmap>

³ <https://github.com/facebookresearch/fastText>

⁴ <https://fasttext.cc/>

the EUR-Lex database⁵ that includes official documents and law and legislation-related texts.[6]

The statistics-based method for obtaining translation equivalents from the EUR-Lex Estonian-Slovak parallel corpus computes the probability that the current word pair is a translation equivalent by measuring the logDice association score.[16] This score considers the frequency of the current word pair (the higher frequency, the higher probability of being a translation equivalent) and the frequency of each word separately (the higher frequency, the lower probability of being a translation equivalent).[13]

Another option for inducing translation equivalents is to utilize existing bilingual dictionaries that share a common language. The idea is to connect meanings from two languages through a third, pivot language. The pivot language is usually well-resourced, for instance, English. This offers an alternative for rare language pairs with no parallel data. However, the resulting translation equivalents are often polluted by the pivot language causing incorrect alignments due to the polysemy of the words.

In this paper, we adopted the results from the Estonian-Slovak dictionary [8] that was obtained by merging English-Estonian and English-Slovak dictionaries. The dictionary was manually assessed on randomly sampled 1,000 translation equivalents, and the achieved accuracy with parallel data was around 40%.

3 Experimental Setup

In the training process, we experimented with two types of pre-trained monolingual word embeddings, FastText monolingual word embeddings[9] and SketchEngine monolingual word embeddings.⁶[10]

Pre-trained FastText monolingual word embeddings for Estonian and Slovak contain around 300,000 words, and we included all of them in the models' training. In SketchEngine pre-trained embeddings for both languages were included around 1 million tokens. For our purposes, we worked only the first 300,000 and added embeddings for words with lower ranks that occurred in the evaluation dataset described in Section 4.

Moreover, we trained MUSE and VECMAP models in a supervised (*model-S*), unsupervised (*model-U*) and semi-supervised mode that relies only on identical strings and numerals (*model-I*). FastText model, we trained in a supervised manner only. In the supervised training, we used our manually created word-to-word dataset with around 5,000 word pairs. The training dataset contained only words occurring in both monolingual word embedding files.

4 Evaluation

This section focuses on the data and methodology used in the evaluation process.

⁵ <https://eur-lex.europa.eu/homepage.html>

⁶ <https://embeddings.sketchengine.eu/>

To evaluate the Estonian-Slovak dictionary induced from parallel corpora, we randomly sampled 1,000 translation equivalents with two constraints: the achieved logDice score must be above 10, and the Estonian words' frequency must be above 1,000. This limited our choice to 35,528 translation equivalents. The aim was to eliminate noisy word pairs such as numbers, proper names from other languages, symbols or words from languages other than Estonian or Slovak.

Despite the dataset limitation, the manual evaluation revealed many mistakes. For example, numbers ('558' : '558'), incorrect proper names ('engström' : 'alfonsi'), website links ('vormis' : 'http://eur-lex.europa.eu/en/index.htm'), different language words ('nuts' : 'nuts'). Thus, the resulting accuracy was 16.1%. The result is displayed in Table 3.

To evaluate the cross-lingual embedding models, we utilized a basic Estonian vocabulary word list. We extracted this word list from the Basic Estonian Dictionary⁷ provided by the Institute of the Estonian Language, which covers basic vocabulary aimed mainly at A2 to B1 CEFR learners.

We assigned the frequency to each word based on its occurrences in Estonian National Corpora from 2017. Afterwards, we randomly picked 70 Estonian words with very high frequency and 30 words with low occurrence. The aim was to see how each model performs on high- and low-frequency words.

The metric picked for evaluation is Precision at k ($P @ k$), which is the proportion of the number of correct translation equivalents to the number of all extracted translation equivalents where k is the amount of extracted target words for each source word. [12]

In the evaluation process, we extracted the 10, 5, and 1 nearest neighbour, their position, and their scores from which we computed two relative scores: the difference between the highest and current scores and the ratio between the highest and current scores. We gained 1,000 translation equivalents for 10 nearest neighbours, 500 translation equivalents for 5 nearest neighbours, and 100 translation equivalents for 1 nearest neighbour. Then, we randomly sampled 100 translation equivalents from the first two groups for the manual evaluation. In the group with 1 nearest neighbour, we evaluated all of them. The results are stated in Table 3.

Additionally, we excluded all unknown words. These unknown words arise because they do not occur in the pre-trained monolingual word embeddings in the first place. FastText did not include 4 of the Estonian words we picked for the evaluation, i.e., 'dressid' (soccer jersey), 'kontoritarbed' (office supplies), 'lastevanemad' (parents), 'ujumisriided' (swimwear). SketchEngine contains all of the words from the evaluation dataset.

During the manual control, we labeled each translation equivalent in two categories: correctness, whether the translation equivalent is correct or not, and in the second category, we reasoned our decision. The motivation was to analyze occurred errors. Table 1 summarizes all labels.

⁷ <http://www.eki.ee/dict/psv/>

Table 1: Manual controls’ labels and examples.

Correctness	Label	Example
YES	meanings match	<i>'ammu' : 'dávno'</i> (long time ago)
YES	inflected word form	<i>'demokraatia' : 'democracii'</i> (democracy)
YES	adjective in different grade	<i>'õnnelik' : 'najšfastejši'</i> (the happiest, correct: happy)
YES	near equivalent or synonym	<i>'sõitma' : 'šoférova'</i> (travel/drive)
YES	additional word needed	<i>'kontoritarbed' : 'kancelárske'</i> (office, correct: office supplies)
No	different part of speech	<i>'sõbralik' : 'priateľstvo'</i> (friendship, correct: friendly)
No	antonym	<i>'kiire' : 'pomalé'</i> (slow, correct: fast)
No	number	<i>'kell' : '17.00'</i> (clock)
No	shortcut	<i>'kilo' : 'kg'</i>
No	symbol	<i>'kell' : '+'</i>
No	meanings do not match	<i>'linn' : 'radnica'</i> (city hall, correct: city)

Apart from assessing the quality of the translation equivalents, we looked at the models’ performance on the high- and low-frequency words. We divided the labelled dataset into these two groups and computed their precision separately.

In most cases, models performed worse on low-frequency words; however, there are some exceptions, i.e., FastText model regardless the monolingual word embeddings, etc.

Furthermore, we observed big gaps between the precision for the high- and low-frequency words in some models. For instance, model MUSE trained with FastText embeddings in a supervised mode, etc. All results are stated in Table 2.

Table 2: The comparison of the precision P@10, P@5, and P@1 when separating words into high- and low-frequency words.

(high-/low-frequency)	P@10	P@5	P@1
FastText			
MUSE-S (%)	26.86/12.12	40/8	58.57/30
MUSE-I (%)	28.76/7.4	25/16.66	45.71/13.33
MUSE-U (%)	27.27/30.43	43.05/32.14	60/26.66
VECMAP-S (%)	43.66/17.24	45.2/29.62	74.28/43.33
VECMAP-I (%)	28.04/22.22	46.05/37.5	60/33.33
VECMAP-U (%)	28.57/20	36.23/41.93	61.42/ 30
FastText (%)	35.21/17.24	33.8/27.58	72.85/40
SketchEngine			
MUSE-S (%)	47.05/34.37	39.39/41.17	70/66.66
MUSE-I (%)	28.35/27.27	38.15/29.16	68.57/56.66
MUSE-U (%)	30.12/23.52	48.64/38.46	71.42/53.33
VECMAP-S (%)	39.70/12.5	48.57/46.66	74.28/66.66
VECMAP-I (%)	38.88/17.85	47.14/23.33	75.71/63.33
VECMAP-U (%)	40/20	39.72/40.74	77.14/60
FastText (%)	13.88/21.42	45.45/47.05	77.14/56.66

After the assessment of the translation equivalents, we visualized scores and positions of the correct and incorrect translation equivalents in 6 different graphs: absolute score and position, absolute score and relative scores (difference, ratio), relative scores and position, and finally, relative scores against each other. The graphs of the VECMAP trained in a supervised mode with FastText word embeddings are displayed in Fig. 1.

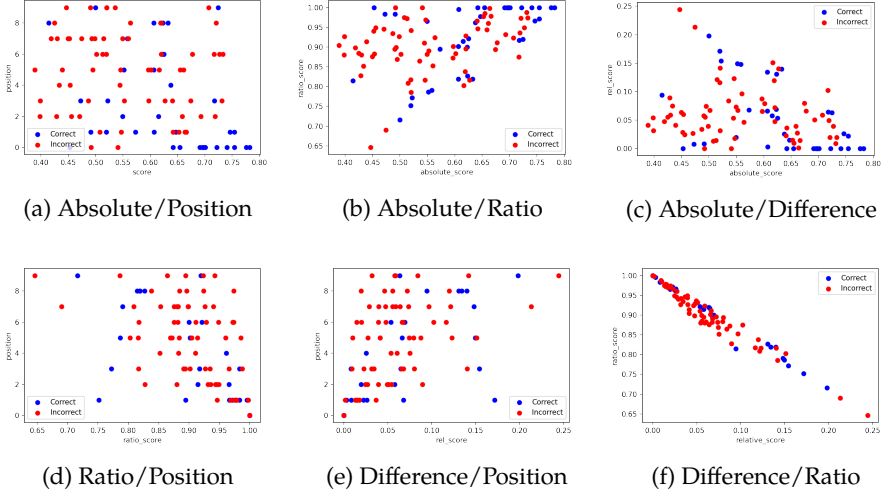


Fig. 1: Various graphs for correct and incorrect translation equivalents extracted from VECMAP trained in a supervised mode with FastText embeddings

According to these graphs, the score line between correct and incorrect ranges between 0.4 - 0.5. This means that instead of extracting the 10, 5, or 1 nearest neighbours for each Estonian word, we can set the limit based on the current induced word's score and eliminate some incorrect translation equivalents. The limit can be expressed as follows:

$$limit = 0.45 + position * 0.01$$

However, given the Fig. 2 obtained scores with SketchEngine monolingual word embeddings were higher. Therefore, the score line rose to 0.6 - 0.7. In this case, the limit can be formulated like this:

$$limit = 0.65 + position * 0.01$$

In the next step, we decided to restrain the score limit of the translation equivalents, compute precision again, and see how the result changed. The results are shown in Table 3.

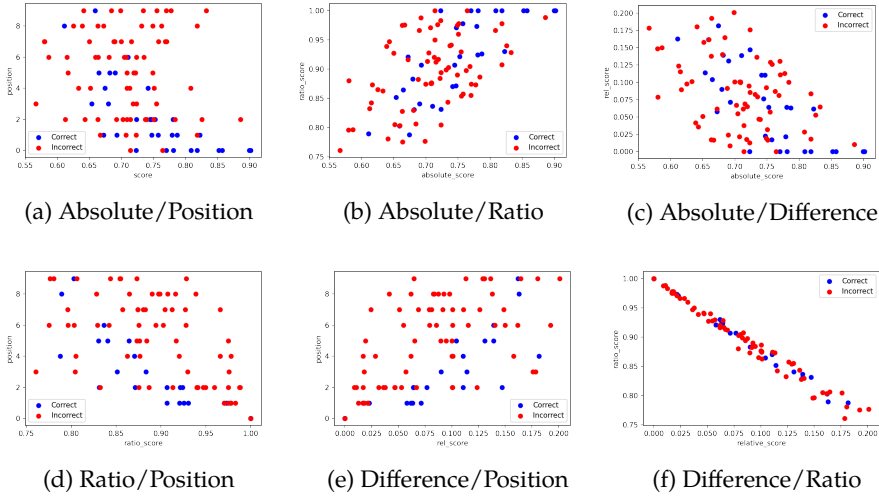


Fig. 2: Various graphs for correct and incorrect translation equivalents extracted from VECMAP trained in a supervised mode with SketchEngine embeddings

Given Table 3, the models’ precision rose significantly after putting a limit constraint. Moreover, models trained with SketchEngine monolingual embeddings performed with and without limit in most cases better than with FastText embeddings.

Generally, all models achieved the best precision when only the closest nearest neighbour was considered. VECMAP trained with SketchEngine monolingual word embeddings was able to ascend the precision up to 72%, which makes it the best model at P@1.

However, some inconsistencies among the results of the models’ precision occurred. The reasons could be various. For instance, the random sampling picked more word pairs with a higher position, the model found better equivalents on higher positions, or we did not set the limit for extracting word pairs accurately.

The most remarkable gap between the monolingual word embeddings was in the model MUSE-S and the FastText model.

MUSE-S trained with SketchEngine embeddings found 30 word pairs that the model trained with FastText embeddings did not find. Reversely, FastText found 6 word pairs that were not in SketchEngine, and both matched in 11 word pairs. Table 4 displays some examples.

FastText trained with FastText monolingual word embeddings found 24 word pairs, SketchEngine 11, and both matched in 11 word pairs. In Table 5 are provided some examples.

Compared to the parallel data-based methods, the pivot dictionary significantly surpassed the Estonian-Slovak dictionary induced from a parallel corpus and is still a concurrence to the cross-lingual embedding models. On the other

Table 3: The precision P@10, P@5, and P@1 of comparable data-based models (MUSE, VECMAP, FastText) before and after applying a limit for the extraction of the translation equivalents compared to the performance of the parallel data-based methods.

	P@10/ Limit	P@5/ Limit	P@1
Comparable data			
FastText			
MUSE-S (%)	22/ 36.84	32/ 45.94	50
MUSE-I (%)	23/ 35.71	23/ 35.29	36
MUSE-U (%)	28/ 34.48	40/ 46.05	50
VECMAP-S (%)	36/ 45.2	41/ 46.06	65
VECMAP-I (%)	27/ 31.57	44/ 50.60	52
VECMAP-U (%)	26/ 32.81	38/ 42.35	52
FastText (%)	30/ 40.9	32/ 48.83	63
SketchEngine			
MUSE-S (%)	43/ 47.16	40/ 49.12	69
MUSE-I (%)	28/ 33.33	36/ 54.54	65
MUSE-U (%)	29/ 33.33	46/ 52.72	66
VECMAP-S (%)	31/ 35.36	48/ 52.17	72
VECMAP-I (%)	33/ 37.5	40/ 47.76	72
VECMAP-U (%)	33/ 33.76	40/ 58.33	72
FastText (%)	16/ 21.81	46/ 52.56	71
Parallel data			
Pivot dictionary (%)	40	-	-
Parallel corpus (%)	16.1	-	-

Table 4: Comparison of the word pairs that were found or were not found by MUSE either trained with FastText (MUSE-S-F) or SketchEngine (MUSE-S-S).

	ET	SK	Pos.	Score	Rank	Correct
MUSE-S-F	<i>laupäev</i>	<i>piatok</i>	1	0.621176	34506	No
	<i>laupäev</i>	<i>sviatok</i>	8	0.578794	34506	No
	<i>suhkur</i>	<i>cukry</i>	1	0.962491	28078	Yes
	<i>samuti</i>	<i>rovnako</i>	5	0.500055	108	Yes
MUSE-S-S	<i>laupäev</i>	<i>nedel'u</i>	4	0.756580	14506	Yes
	<i>laupäev</i>	<i>Nedela</i>	8	0.733683	14506	Yes
	<i>laupäev</i>	<i>víkend</i>	5	0.756557	14506	No
	<i>suhkur</i>	<i>škrob</i>	6	0.802334	7490	No
	<i>samuti</i>	<i>rovnako</i>	4	0.792155	190	Yes

Table 5: Comparison of the word pairs that were found or were not found by FastText either trained with FastText (FastText-F) or SketchEngine (FastText-S).

	ET	SK	Pos.	Score	Rank	Correct
FastText-F	<i>laul</i>	<i>pesnička</i>	0	0.253858	752	Yes
	<i>lõplik</i>	<i>konečné</i>	4	0.214931	5056	Yes
	<i>õnnelik</i>	<i>milovaný</i>	1	0.172903	9829	No
	<i>sõitma</i>	<i>cestovať</i>	0	0.264190	13698	Yes
	<i>sõitma</i>	<i>jazda</i>	2	0.180665	13698	No
FastText-S	<i>laul</i>	<i>hymna</i>	7	0.573021	4021	No
	<i>lõplik</i>	<i>presný</i>	8	0.522405	6906	No
	<i>õnnelik</i>	<i>šťastné</i>	6	0.517561	2381	Yes
	<i>sõitma</i>	<i>vieźć</i>	3	0.628647	2734	Yes

hand, the parallel corpus-based method performed noticeably worse than most of the cross-lingual embedding models.

Importantly, we did not focus on words' senses and recall of the models. As we can see from the examples, the cross-lingual models could connect various word forms, but they perform poorly on different word senses. If we focused on models' recall, the parallel data-based models would perform better as they can capture context information.

5 Conclusion and Future Work

We have compared the precision of the translation equivalents induced by three approaches utilizing comparable data with two parallel data-based approaches. We have manually analysed the obtained translation equivalents and have provided insight into occurred errors. Additionally, we have introduced a new formula for extracting translation equivalents from cross-lingual embedding models more effectively.

Although the parallel data are still a competition to the comparable data, as they contain rich context information, in some disciplines, the comparable data outperformed parallel data significantly. Moreover, given the amount of research conducted in the cross-lingual embedding models' field, they represent a good alternative and show promising results for the future, either stand-alone or as supplement data, especially for low-resource languages or rare language pairs.

Finally, the formula for extracting translation equivalents was inferred manually based on the graphs' observations. However, every model and monolingual word embeddings are specific and require different weights for their limit. We propose for future work to implement an algorithm that would tailor the most appropriate limit for each model separately.

Acknowledgments The research in this paper was supported by the Internal Grant Agency of Masaryk University, project MUNI/IGA/1285/2021.

References

1. Artetxe, M., Labaka, G., Agirre, E.: Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 2289–2294 (2016), <https://aclanthology.org/D16-1250>
2. Artetxe, M., Labaka, G., Agirre, E.: Learning bilingual word embeddings with (almost) no bilingual data. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 451–462 (2017), <https://aclanthology.org/P17-1042>
3. Artetxe, M., Labaka, G., Agirre, E.: Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence. pp. 5012–5019 (2018)
4. Artetxe, M., Labaka, G., Agirre, E.: A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 789–798 (2018), <https://arxiv.org/abs/1805.06297>
5. Atkins, B., Rundell, M.: The Oxford Guide to Practical Lexicography. OUP Oxford (2008)
6. Baisa, V., Michelfeit, J., Medved', M., Jakubíček, M.: European Union language resources in Sketch Engine. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16). pp. 2799–2803. European Language Resources Association (ELRA), Portorož, Slovenia (May 2016), <https://aclanthology.org/L16-1445>
7. Conneau, A., Lample, G., Ranzato, M., Denoyer, L., Jégou, H.: Word translation without parallel data. arXiv preprint arXiv:1710.04087 (2017), <https://arxiv.org/abs/1710.04087>
8. Denisova, M.: Compiling an estonian-slovak dictionary with english as a binder. In: Electronic lexicography in the 21st century. Proceedings of the eLex 2021 conference. pp. 107–120 (2021)
9. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)
10. Jakubíček, M., Kilgarrieff, A., Kovář, V., Rychlý, P., Suchomel, V.: The tenten corpus family. In: 7th International Corpus Linguistics Conference CL 2013. pp. 125–127. Lancaster (2013), <http://ucrel.lancs.ac.uk/cl2013/>
11. Joulin, A., Bojanowski, P., Mikolov, T., Jégou, H., Grave, E.: Loss in translation: Learning bilingual word mapping with a retrieval criterion. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (2018)
12. Kementchedjhieva, Y., Hartmann, M., Søgaard, A.: Lost in evaluation: Misleading benchmarks for bilingual dictionary induction. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 3336–3341. Association for Computational Linguistics, Hong Kong, China (Nov 2019). <https://doi.org/10.18653/v1/D19-1328>, <https://aclanthology.org/D19-1328>

13. Kovář, V., Baisa, V., Jakubíček, M.: Sketch Engine for Bilingual Lexicography. *International Journal of Lexicography* **29**(3), 339–352 (07 2016). <https://doi.org/10.1093/ijl/ecw029>, <https://doi.org/10.1093/ijl/ecw029>
14. Lample, G., Conneau, A., Denoyer, L., Ranzato, M.: Unsupervised machine translation using monolingual corpora only. arXiv preprint arXiv:1711.00043 (2017), <https://arxiv.org/abs/1711.00043>
15. Ruder, S., Vulić, I., Søgaard, A.: A survey of cross-lingual word embedding models. *J. Artif. Int. Res.* **65**(1), 569–630 (May 2019), <https://doi.org/10.1613/jair.1.11640>
16. Rychlý, P.: A lexicographer-friendly association score. In: RASLAN (2008)

Automatic Identification of Speakers and Parties in Steno Protocols of the Czech Parliament

Ota Mikušek

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
`xmikusek@fi.muni.cz`

Lexical Computing, Brno, Czech Republic
`ota.mikusek@sketchengine.eu`

Abstract. There are many methods of machine learning. This paper shows an application of basic machine learning methods like bag of words, random forest and naive Bayes on classification task of assigning sentences to members and parties of the Czech Parliament.

Keywords: scikit-learn, embedding, SVM, random forest, naive Bayes, n-gram, CountVectorizer, classification

1 Introduction

Based on the data from the Czech Parliament between the years 2015 and 2019 [1], containing over 1,400,000 sentences of 14 political parties and over 100,000 tokens for 60 members. This work compare the methods of machine learning that can classify the individual sentences of different members with more than 100,000 spoken tokens and sentences of different parties of the Czech Parliament. Among the methods for analysis there are: Decision trees, K-nearest neighbors, Bag of Words, Naive Bayes and others, or similar. Evaluation of these methods will be done by covering 20% of source data for all models. For some models input in form of word embeddings was chosen. Best models have been tuned and evaluated.

All source code used for the implementation and evaluation is open source¹.

2 Data exploratory analysis

The data [1] was obtained in a ZIP file containing the TEI, conll, and vertical formats. Manual inspection of a random sample of 500 sentences did not reveal any errors in typography or technical formatting. In all formats, words, lemmas, tags, and position of words in a sentence were already precomputed.

¹ <https://gitlab.fi.muni.cz/xmikusek/czech-parlament-prediction>

3 Data preprocessing

To ease the development and unify annotation schemes of all the datasets, all data was reprocessed using Unitok [5] for new tokenization, Desamb [8] for part-of-speech tagging and lemmatization and the SET tool for parsing [4]. Original annotations were not used. Data were split into three parts, test data (20 %), validation data (16 %) and training data (64 %) with balanced representation of every member sentences amount. In all models python library scikit-learn [6] was used, expect models that used word embeddings as input, these models we also using FastText [3] with external precomputed word embeddings [2].

3.1 Sentences

Since most machine learning methods always expect the same input length, sentences were transformed with the scikit-learn CountVectorizer, which can count n-grams in a sentence and transform them into numeric vectors of the same length (the resulting vector, however, is not unique for each sentence). Vectors for n-grams of lengths 1 to 3 were created simultaneously.

3.2 Lemmatization

Similar to sentences, the input for methods needs to be the same length. The solution is the same as in sentences. We use sklearn CountVectorizer to create vectors from n-grams of lengths 1 to 3.

3.3 Tagging

For tagging, the same problem arose as above, the problem that most machine learning methods expect each sentence to have the same size of resulting tags. Since desamb creates tags as pairs of characters, where the first describes the tags and the second the value, this problem was solved by preprocessing during learning by concatenating all the tags and then splitting the characters into pairs. A vector (of uniform length) representing the number of occurrences of individual pairs was subsequently created from these pairs. As a result, the information with which word the tag is associated was lost.

3.4 Syntactic analysis

Similar to tagging, the problem that arose is that most machine learning methods expects each sentence to have the exact size of the resulting analysis. The problem was solved by creating ordered triplets word1, dependency, and word2. Subsequently, n-grams of lengths 1 to 3 were created simultaneously through sklearn CountVectorizer.

3.5 Additional information

The number of words in the sentence and the number of characters from the set {".", ",", "!", "?", "-", "/", "\""} [7] were added as additional information for classification.

4 Classification results

Four types of models were trained (Table 1, Table 2, Table 3, Table 4, Table 5):

1. only from lemmatized sentence
2. only from lemmatized sentence with balanced classes
3. on all parameters
4. on all parameters with balanced classes

On two tasks:

1. party classification
2. member classification

Each table contains identifier of model, that was used as a folder name containing that model, and model evaluation on validation or test data.

5 Models

5.1 Baseline (Simple bag of words)

This model is only exception and was trained on sentence word vectors (instead of lemmas) with scikit-learn TreeClassifier to establish baseline.

This model achieved a precision of 29 % and a recall of 9 % on validation tests in the party classification (Table 1). There are only two parties with higher recall. First with recall 34 % and second with recall 20 %. The other parties have a recall of less than 14 %.

On the contrary, both precision and recall are surprisingly higher than expected in member classification (Table 3). This could be happening, because some Parliament members use some words that are specific only to them.

5.2 Bag of words lemmatized

This model was created again with scikit-learn TreeClassifier, expect input was lemmatized sentences. Model was tested only on validation tests in party classification (Table 1). Compared to baseline, it seems like word form is not important for correct classification.

Table 1: Party classification results on validation tests

Model folder name	precision (%)	Recall (%)	Classes with precision and recall 0 %
embeddings_lsvc	24	25	5
embeddings_naive_bayes	16	21	8
embeddings_random_forest	37	30	1
embeddings_svc	DNF	DNF	DNF
simple_bag_of_words	29	9	3
lemma_bag_of_words	27	28	1
lemma_bag_of_words_limited250_balanced_ngrams1-3	29	24	0
lemma_bag_of_words_limited50	33	27	1
lemma_bag_of_words_limited500	27	28	1
lemma_bag_of_words_limited500_balanced_ngrams1-3	27	25	0
lemma_bag_of_words_limited50_balanced	35	14	0
lemma_bag_of_words_limited50_balanced_ngrams1-3	38	14	0
lsvc	35	35	1
lsvc_all_params	38	38	1
lsvc_all_params_balanced	38	38	1
lsvc_balanced	35	34	0
naive_bayes	49	32	1
naive_bayes_all_params	53	30	5
random_forest	53	26	1
random_forest_all_params	52	28	3
random_forest_all_params_balanced	35	20	1
random_forest_all_params_balanced_limited250	30	18	0
random_forest_all_params_balanced_limited50	30	18	0
random_forest_balanced	31	19	1
simple_bag_of_words	29	9	4
svc	19	18	11
svc_all_params	DNF	DNF	DNF
svc_all_params_balanced	DNF	DNF	DNF
svc_bag	43	29	2
svc_bag_all_params	46	26	5
svc_bag_all_params_balanced	46	26	5
svc_bag_balanced	43	29	5
svc_balanced	19	18	11

Table 2: Party classification on validation tests with tuned parameters

Model folder name	precision (%)	Recall (%)	Classes with precision and recall 0 %
random_forest	53	26	1
random_forest_tuning_1	53	26	1
random_forest_tuning_2	52	27	1
random_forest_tuning_3	53	25	3
random_forest_tuning_4	53	26	1
random_forest_tuning_5	54	26	1
random_forest_tuning_6	53	26	1
random_forest_tuning_7	51	27	1
random_forest_tuning_8	51	27	1
random_forest_tuning_9	54	26	1
random_forest_tuning_10	51	27	1

Table 3: Member classification results on validation tests

Model folder name	precision (%)	Recall (%)	Classes with precision and recall 0 %
tokens_100000_embeddings_knn	31	29	0
tokens_100000_embeddings_ISVM	23	16	0
tokens_100000_embeddings_naive_bayes	12	15	38
tokens_100000_embeddings_random_forest	30	26	0
tokens_100000_embeddings_SVM	33	26	0
tokens_100000_naive_bayes_all_params	44	22	38
tokens_100000_random_forest	43	20	28
tokens_100000_random_forest_all_params	43	21	30
tokens_100000_random_forest_all_params_balanced	35	23	0
tokens_100000_random_forest_balanced	32	20	0
tokens_100000_simple_bag_of_words	26	27	0
tokens_100000_svc	42	33	0
tokens_100000_svc_all_params	DNF	DNF	DNF
tokens_100000_svc_all_params_balanced	DNF	DNF	DNF
tokens_100000_svc_balanced	36	30	0

Table 4: Member classification on validation tests with tuned parameters

Model folder name	precision (%)	Recall (%)	Classes with precision and recall 0 %
tokens_100000_random_forest	43	20	28
tokens_100000_random_forest_tuning_1	45	20	28
tokens_100000_random_forest_tuning_2	47	21	20
tokens_100000_random_forest_tuning_3	43	18	34
tokens_100000_random_forest_tuning_4	45	20	27
tokens_100000_random_forest_tuning_5	42	19	30
tokens_100000_random_forest_tuning_6	47	22	17
tokens_100000_random_forest_tuning_7	46	21	23
tokens_100000_random_forest_tuning_8	47	21	20
tokens_100000_random_forest_tuning_9	45	21	25
tokens_100000_random_forest_tuning_10	45	21	24

Table 5: Classification on test set

Model folder name	precision (%)	Recall (%)	Classes with precision and recall 0 %
random_forest_tuning_5	55	26	1
tokens_100000_random_forest_tuning_6	47	22	16

5.3 Naive Bayes

This model was trained with scikit-learn MultinomialNB. It was expected that, every party will say sentences, that have some specific key phrases just for that party. Naive Bayes proves that this idea is maybe not entirely wrong, with it's increase in successful classification (Table 1) in comparison with baseline.

But on the contrary in member classification 38 members are not even classified once in testing (Table 3). This could mean that some key phrases are shared in members group and therefore these members are hard to distinguish from one another.

5.4 Random forest

In party classification (Table 1), model was trained with scikit-learn RandomForestClassifier. Results were very similar with Naive Bayes, but only 1 class remained with 0 % for precision and recall. Average precision was 53 %, and recall was 26 %. The best result was achieved on only lemmatized sentences. This model was later selected for tuning.

Best model for member classification was created when using all features as input with a precision of 43 % and a recall of 21 % (Table 3). However, 30 classes out of 60 have a precision and a recall of 0 %. The model that received only lemmatized sentences as input had a precision of 43 % and a recall of 20 % and was later chosen for tuning at the cost of a 1 % loss in a recall but addition of 2 classified classes.

When model was tuned (Table 2, Table 4), parameters `min_samples_leaf`, `n_estimators` and `max_depth` of RandomForestClassifier were modified. Best models were evaluated on test data (Table 5).

5.5 SVM (Support vector machine)

Since SVM was running too long, for party classification (Table 1), regularization parameter $C=0.001$ was used to make it run faster at the cost of less successful classification.²

For member classification (Table 3), model was unmodified.

5.6 Bag of SVM

This model was created as the reaction on slow learning SVM with combination of scikit-learn BaggingClassifier and SVC. A bag of 20 SVMs, where each SVM classifies the input and then vote for over all classification. This method was used only in party classification (Table 1).

² See <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVM.html> for details.

5.7 Naive Bayes with word embeddings

Input for this model was in form of word embeddings processed with FastText `get_vector` function. Precision of 24 % and a recall of 25 % in the party classification. 5 classes out of 14 had precision and recall of 0 %. For the member classification, a precision of 37 % and a recall of 32 % were achieved. 4 classes out of 60 had a precision and a recall of 0 %.

5.8 Random forest with word embeddings

Input for this model was in form of word embeddings processed with FastText `get_vector` function. The model did not surpass the previous models in party classification. Precision 37 % and recall 30 %. 1 class out of 14 had precision and recall of 0 %.

A precision of 43 % and a recall of 20 % were achieved in member classification. 28 classes out of 60 had precision and recall of 0 %.

5.9 SVM with word embeddings

Input for this model was in form of word embeddings processed with FastText `get_vector` function. Unfortunately, the learning did not end in a reasonable time, but the version with a linear kernel did with a precision of 24 % and a recall of 25 % in the party classification. 5 classes out of 14 had precision and recall of 0 %. For the member classification, a precision of 37 % and a recall of 32 % were achieved. 4 classes out of 60 had a precision and a recall of 0 %.

5.10 KNN with word embeddings

Input for this model was in form of word embeddings processed with FastText `get_vector` function. Only used in the member classification, with resulting precision of 31 % and recall of 29 %. No class out of 60 has a precision and recall of 0 %.

6 Conclusions

Models were compared using validation data based on precision, recall, and the number of classes remaining in the model with precision and recall at 0 %. In both tasks, assigning a sentence to a party and assigning a sentence to one of the 60 Parliament members with more than 100,000 spoken tokens, the random forest model was the most successful, which, after the final tuning, achieved a precision of 55 %, recall 26 % and only one class failed to classify at all in party classification. In the member classification, the random forest model after the final tuning achieved a precision of 47 %, a recall of 22 %, and had a problem classifying 16 people out of 60.

The resulting models alone are not suitable for this type of sentence classification. In particular, checking whether the sentence could have been uttered by a specific politician is not suitable due to the low precision in both tasks, which is around 50 %.

7 Future work

7.1 Data expansion

Since all sentences have the same political topic, it is hard to find differences between one and another. More data could give models more rare words to work with.

7.2 Comparison with transformers or neural networks

The current trend in machine learning is transformers. It might be interesting to see how the Czer model for sentence classification or perhaps models using the gpt2 would deal with this task.

7.3 Alternative text segmentation

Machine learning may not be able to find differences in texts when they are all about the same political topic. It is a question of whether it is even possible to achieve good results based only on the analysis of one sentence.

Analysis at the level of entire paragraphs or documents would allow obtaining more features from the input, such as the number of sentences spoken, frequent repetition or use of rich vocabulary, or average sentence length.

References

1. Erjavec, T., Ogrodniczuk, M., Osenova, P., Ljubešić, N., Simov, K., Grigorova, V., Rudolf, M., Pančur, A., Kopp, M., Barkarson, S., Steingrímsson, S., van der Pol, H., Depoorter, G., de Does, J., Jongejan, B., Haltrup Hansen, D., Navarretta, C., Calzada Pérez, M., de Macedo, L.D., van Heusden, R., Marx, M., Çöltekin, Ç., Coole, M., Agnoloni, T., Frontini, F., Montemagni, S., Quochi, V., Venturi, G., Ruisi, M., Marchetti, C., Battistoni, R., Sebők, M., Ring, O., Dargis, R., Utko, A., Petkevičius, M., Briedienė, M., Krilavičius, T., Morkevičius, V., Bartolini, R., Cimino, A., Diwersy, S., Luxardo, G., Rayson, P.: Linguistically annotated multilingual comparable corpora of parliamentary debates ParlaMint.ana 2.1 (2021), <http://hdl.handle.net/11356/1431>, slovenian language resource repository CLARIN.SI
2. Herman, O.: Precomputed word embeddings for 15+ languages. In: RASLAN. pp. 41–46 (2021)
3. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
4. Kovář, V., Horák, A., Jakubíček, M.: Syntactic analysis using finite patterns: A new parsing system for czech. In: Human Language Technology. Challenges for Computer Science and Linguistics. pp. 161–171. Springer, Berlin/Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-20095-3_15
5. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text tokenisation using unitok. In: Horák, A., Rychlý, P. (eds.) RASLAN 2014. pp. 71–75. Tribun EU, Brno, Czech Republic (2014)

6. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011)
7. Sari, Y., Stevenson, M., Vlachos, A.: Topic or style? exploring the most useful features for authorship attribution. In: *Proceedings of the 27th International Conference on Computational Linguistics*. pp. 343–353. Association for Computational Linguistics, Santa Fe, New Mexico, USA (Aug 2018), <https://aclanthology.org/C18-1029>
8. Šmerk, P.: *Unsupervised Learning of Rules for Morphological Disambiguation*. In: *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg (2004)

Keyness Analysis and Its Representation in Russian Academic Papers on Computational Linguistics: Evaluation of Algorithms

Maria Khokhlova 
and Mikhail Koryshev 

St Petersburg State University
Universitetskaya emb. 7-9-11,
199034 St Petersburg, Russia
m.khokhlova@spbu.ru, m.koryshev@spbu.ru

Abstract. Extraction of relevant lexis has gained significance as the amount of information is continuously growing with news, posts on social networks, reviews, academic papers, etc. piling up. Automated algorithms are needed to analyze texts to facilitate understanding of their content. The paper scrutinizes methods for keyword extraction in abstracts of Russian scientific texts on computational linguistics. Unsupervised algorithms based on statistics, graphs and machine learning principles are considered. The results are evaluated against the keywords assigned by authors themselves, followed by expert opinion. Log-likelihood produced the best results in comparison with author keywords, while KeyBERT implementation with vectorizers outperformed other algorithms according to expert assessment.

Keywords: Keyword extraction, Academic papers, Abstracts, Computational linguistics, Log-likelihood, TextRank, RAKE, YAKE, KeyBERT

1 Introduction

Keyword extraction has never been more relevant. Continuous increase in information volume makes it difficult for users to familiarize with all the emerging data. It is impossible to read all the papers on a particular topic or all the news. This challenge can be partly resolved by automated methods that allow us to grasp the main content of any texts.

Our study focuses on academic texts and tools to extract significant and meaningful phrases, in particular. We explore a number of methods and evaluate them using keywords tagged by authors or selected manually by experts.

The paper has the following structure: Section 2 provides a brief overview of related studies; Section 3 describes data and presents methods and relevant notions; Section 4 examines the results, followed by Section 4 that concludes the paper and outlines future perspectives.

2 Background Research

The term “key word” refers to “a word which occurs with unusual frequency in a given text” [1, p. 236], meaning that its frequency is unusual compared to a large reference corpus. Extraction of relevant words and phrases from texts is closely related to the tasks of computational linguistics dealing with automatic term and collocation extraction or named entity recognition. We can say that, to a certain extent, the methods that are used to solve them intersect (chi-squared, log-likelihood ratio [2]).

The simplest way to find significant phrases in a document is to make a list of n-grams of either lemmas or word forms ranging them according their occurrences. There is a class of measures based on the comparison between reference and focus (or domain-specific) corpora. This statistical approach was implemented in Wordsmith Tools [3] and then further applied in a number of dictionaries and software systems. Sketch Engine supports keyword and term extraction using its own score to compare frequencies of single word or multiword units in focus and reference corpora [4,5]. Depending on a specific parameter, the measure favors either low-frequency words (with high keyness and thus highly relevant for a focus corpus), or high-frequency words (with low keyness). Similarly, AntConc weights candidates in different corpora that a user can upload [6].

More elaborate statistical methods involve calculating frequencies not just in one document, but in a collection of them (for example, TF-IDF). The same principles underlie KPMiner [7] that additionally filters n-grams. The RAKE algorithm (Rapid Automatic Keyword Extraction) was introduced in [8] and extracts multilingual keywords represented by n-grams. It is characterized by assigning more weight to longer sequences of words. The algorithm applies stop words and a delimiter list calculating statistics to search for multi-word terms [ibid]. Based on frequencies and count of relations between words and phrases, the method estimates the weight for each candidate and ranks them according to the values. YAKE is a corpus- and language-independent algorithm that employs a mixture of linguistic and statistical features such as casing, word position, relatedness to context, frequency, and dispersion of lexical items in different sentences [9,10]. Automatic extraction with YAKE is based upon the assumptions about the behavior of terms in documents. Relevant keywords are supposed to be concentrated more at the beginning of a document. Or a large number of different terms that co-occur with the candidate word can be crucial to indicate its meaningless character.

Graph-based ranking methods have been successfully used in a number of applications, keyword extraction being one of them. TextRank [11] was proposed for two language processing procedures, namely, unsupervised keyword and sentence extraction. It ranks keyword candidates according to their positions in graphs. One of the recent algorithms is RaKUn [12] that merges similar words into meta-vertices, reducing the number of vertices, as well as edges. It computes load centrality measure that is based on the number of shortest paths for a given vertex and thus estimates the importance of vertices in graphs (i.e.

keywords). Most recent and state-of-the-art approaches belong to machine learning. Methods based on transformers seem to be the most promising ones, BERT being one of them. Its modification KeyBERT belongs to embedding-based methods that use word distributions and sentence representations. It was proposed in [13] and is based on the bidirectional pretrained BERT model. Candidate keyphrases are ranked according to the cosine similarity.

Keywords “capture” the essence of texts and thus their extraction from academic papers is in a focus of attention in a number of works. In [14], the authors extract noun phrases in scientific abstracts in English (the Inspec dataset) based on pos-tags to use the results in academic search systems. The authors evaluate approaches that allow them to weight candidate phrases and then apply the metrics to rank them in terms of average geometric mean, pointwise mutual information, tf-idf, and entropy-based measures. Bruches et al. [15] study methods for entity recognition and relation extraction applied to Russian texts on information technologies. The authors collected a corpus of abstracts and annotated manually fragments with terms (about 2,000 items) represented by noun phrases and semantic relations (620 items involving “cause”, “compare”, “isa”, “partof”, “synonyms” and “usage”). Nguyen & Zaslavskiy [16] deal with keyphrase extraction in papers written in Russian and English using sentence embeddings. They propose a supervised learning model that calculates scores estimating the quality of every keyword. LanAKey_Ru was proposed in [17] for keyword extraction in Russian papers on mathematical modeling. Based on n-grams, the algorithm employs stop lists for their filtering and evaluates relevance of noun phrases using statistical and linguistic features.

3 Data and Methodology

3.1 Data Collection

We collected a corpus of abstracts in Russian for the papers submitted to “Dialogue” (from 2017 to 2022). It is the largest conference on computational linguistics and intellectual technologies that focuses on the Russian language and is held annually in Russia [18].

The corpus of abstracts comprises about 27,000 words and contains texts of different lengths (as authors do not always follow the template). We encountered, oddly enough, a certain challenge in collecting texts in Russian: many high-ranked professional conferences in Russia pursue a widest possible audience, as well as indexing in international databases, and hence the majority of talks is given in English. Therefore, most articles are submitted in English, as well as texts published in proceedings. However, papers may contain abstracts in the Russian language upon authors’ consideration.

3.2 Methods

In our study, we deal with a number of unsupervised methods for keyword extraction as they require no labeled training data. These methods rely on

statistics, embeddings, and graphs representing different approaches. Statistical measures involved joint frequency (chosen as baseline) and log-likelihood. Among other approaches we used YAKE, RAKE, TextRank, and KeyBERT. Preprocessing included stop words removal, as well as lemmatization and morphological annotation that were performed with pymorphy2 [19].

The quality of the first 100 candidates was evaluated in two ways: by comparing with author keywords (a predefined set of terms assigned by the authors themselves) and by expert evaluation.

4 Results

4.1 Author Keywords

Specialized dictionaries can be used for evaluation as a benchmark. For example, dictionaries of linguistics terminology. Dictionary by Akhmanova [20] is a recognized source for Russian, though unsuitable for such a rapidly developing field as computational linguistics due to a broad linguistic scope, on the one hand, and outdated material, on the other.

In our evaluation we consider the keywords that are given in the papers and were attributed by the authors themselves. This predefined set of terms is compared to extracted candidate keywords. In total, we collected 822 author terms. The most frequent ones are: *BERT* (14), *klassifikacija tekstov* ‘text classification’ (6), *korpus* ‘corpus’ (6), *korpusnaja lingvistika* ‘corpus linguistics’ (11), *lemmatizacija* ‘lemmatization’ (7), *morfologicheskij analiz* ‘morphological analysis’ (7), *nejronnye seti* ‘neural networks’ (6), *rechevoj korpus* ‘spoken corpus’ (5), *russkij jazyk* ‘Russian language’ (42), *semantika* ‘semantics’ (7). Among the examples we find both general linguistic terms and highly specialized ones that are typical for computational linguistics.

The authors assign keywords inconsistently and in their own way. The analysis revealed synonyms in the lists of keywords. For example, *vybor zagolovkov* ‘choice of titles’ vs *generacija zagolovkov* ‘title generation’, *generacija zagolovkov* ‘title generation’ vs *generacija novostnyh zagolovkov* ‘news title generation’, *summarizacija* ‘summarization’ vs *summarizacija tekstov* ‘text summarization’, *predobuchennye modeli* ‘pretrained models’ vs *predobuchennye jazykovye modeli* ‘pretrained language models’, *diskursivnye markery* ‘discourse markers’ vs *diskursivnye slova* ‘discourse words’, *semanticheskaja blizost* ‘semantic similarity’ vs *semanticheskaja blizost tekstov* ‘semantic similarity of texts’. Different word forms within the same node term are identified, e.g. singular vs plural (*generacija teksta* ‘generation of text’ vs *generacija tekstov* ‘generation of texts’). Shortenings and standard forms represent another example of using the same terms, e.g. *avtomaticheskaja morfologicheskaja razmetka* ‘automatic morphological analysis’ vs *avtomaticheskaja morforazmetka* ‘automatic morphoanalysis’, *avtomaticheskoe referirovanie tekstov* ‘automatic summarization of texts’ vs *avtoreferirovanie tekstov* ‘autosummarization of texts’.

In computational linguistics, a large number of terms come from the English language, so in some cases we can find a transliteration of terms (for example,

gepping ‘gapping’, *embeddingi* ‘embeddings’), and in some cases, duplication of existing ones (for example, *evaluacija* ‘evaluation’ instead of *ocenka* ‘evaluation’, *simplifikacija* ‘simplification’ instead of *uproshhenie* ‘simplification’). On the one hand, this may indicate that there is no established term, or the authors are influenced by their English-based scientific background and want to clarify what material the study is being conducted on, as well as indicate certain methods and separate their studies from previous ones. On the other hand, this inconsistency could be eliminated if an automatic system were used that would allow the selection of suitable words or phrases from a precompiled list.

4.2 N-grams and joint frequency

The most frequent bigram candidate terms in the “Dialogue” corpus of abstracts include (frequencies are given in parentheses): *russkij jazyk* ‘Russian language’ (134), *nabor dannyh* ‘data set’ (34), *jazykovaja model* ‘language model’ (31), *imenovannaja sushhnost* ‘named entity’ (26), *estestvennyj jazyk* ‘natural language’ (18), *nacional’nyj korpus* ‘national corpus’ (16), *semanticheskij sdvig* ‘semantic shift’ (15), *rechevoj akt* ‘speech act’ (15), *vektornoe predstavlenie* ‘word embedding’ (13), *mashinnoe obuchenie* ‘machine learning’ (12), *znachenie slova* ‘word meaning’ (12), *nejronnaja set* ‘neural network’ (11), *baza dannyh* ‘database’ (9), *semanticheskij sketch* ‘semantic sketch’ (8), *morfologicheskij analiz* ‘morphological analysis’ (7), *diskursivnoje slovo* ‘discourse marker’ (7), *mehanizm vnimanija* ‘attention mechanism’ (7), *rechevoj sbor* ‘speech failure’ (7), *individual’noje razlichije* ‘individual difference’ (7), *kljuchevoje slovo* ‘keyword’ (6).

The most typical frequency lexemes in abstracts are: *jazyk* ‘language’ (266), *model* ‘model’ (214), *russkij* ‘Russian’ (212), *tekst* ‘text’ (211), *korpus* ‘corpus’ (192), *zadacha* ‘task’ (177), *stat’ja* (163) ‘paper’, *rezul’tat* ‘result’ (149), *slovo* ‘word’ (146), *metod* ‘method’ (128), *rabota* ‘work’ (116), *dannye* ‘data’ (115), *issledovanie* ‘study’ (111), *znachenie* ‘meaning’ (103), *sorevnovanie* ‘competition’ (94), *kachestvo* ‘quality’ (94), *semanticheskij* ‘semantic’ (93), *osnova* ‘base’ (80), *podhod* (77) ‘approach’, *tip* ‘type’ (75).

We also outlined typical trigrams: *korpus russkogo jazyka* ‘corpus of Russian’ (17), *obrabotka estestvennogo jazyka* ‘natural language processing’ (9), *raspoznavanie imenovannyh sushhnostej* ‘named entities recognition’ (8), *nositel’ russkogo jazyka* ‘speaker of Russian’ (6), *predobychennye jazykovye modeli* ‘pre-trained language models’ (5), *metody mashinnogo obuchenija* ‘machine learning methods’ (4), *vektornye predstavlenija slov* ‘word embeddings’ (4), *ponimanie estestvennogo jazyka* (4) ‘natural language understanding’, *upotreblenie roditel’nogo partitivnogo* (4) ‘usage of partitive genitive’, *rekurrentnye nejronnye seti* ‘recurrent neural networks’ (3), *izmenenie znachenija slova* ‘change in word meaning’ (3), *verbal’naja reakcija slushajushhego* ‘hearer’s verbal response’ (3), *Odin rechevoj den* ‘One speaker’s day’ (a title of the project) (3), *semanticheskaja slozhnost’ slova* ‘semantic complexity of words’ (3), *obnaruzhenie semanticheskikh sdvigov* ‘semantic shift detection’ (3).

Most of the top-list for bigrams and trigrams do reflect the terminology of computational linguistics, while the list for unigrams reveal broader linguistic and science terms.

4.3 Log-likelihood

For log-likelihood measure, the output almost completely coincides with n-grams mentioned above, with a difference in ranking only: *russkij jazyk* ‘Russian language’, *nabor dannyh* ‘data set’, *jazykovaja model* ‘language model’, *rechevoj akt* ‘speech act’, *semanticheskij sdvig* ‘semantic shift’, *nacional’nyj korpus* ‘national corpus’, *vektornoe predstaolenie* ‘word embedding’, *nejronnaja set* ‘neural network’, *estestvennyj jazyk* ‘natural language’, *mashinnoe obuchenie* ‘machine learning’, *individual’noje razlichije* ‘individual difference’, *fonovoe znanie* ‘background knowledge’, *mehanizm vnimanija* ‘attention mechanism’, *kommunikativnaja neudacha* ‘communication failure’, *imenovannaja sushhnost* ‘named entity’, *roditel’nyj partitivnyj* ‘partitive genitive’, *rechevoj sboj* ‘speech failure’, *izvlechenije otnoshenij* ‘relation extraction’, *predmetnaja oblast* ‘subject area’, *semanticheskij sketch* ‘semantic sketch’. The measure achieved the best score for the top list of candidates (however, with bigrams only), when evaluated against author keywords, outperforming other methods.

4.4 YAKE

Opposed to other algorithms, YAKE ranks candidate terms in ascending order, i.e. the lower the score, the more relevant the keyword is. The algorithm outperformed the above-mentioned two methods by suggesting unigrams, bigrams and trigrams as candidates. Among the first 100 candidates, we found more than 50 percent represented by verb phrases and simple clauses (e.g. *stat’ja predstavljajet rezul’taty* ‘the paper presents the results’, *ispol’zovat’ korpus tekstov* ‘to use a text corpus’, *predstavljat’ rezul’taty sovnovanija* ‘to present competition results’, *dannaja rabota posvjashhena* ‘the work deals with’, *rezul’taty pokazala model* ‘the model showed results’). This may indicate that YAKE can be used, for example, for summarization and similar tasks, since it extracts prefabricated and frequent chunks.

4.5 RAKE

We used *multi-rake* implementation [21] that supports Russian texts with minimum frequency for keywords equal to 2. The following candidates were extracted by implementing the algorithm: *obnaruzhenie novostnyh sobytij* ‘event detection from news’, *morfologicheskij bogatij jazyk* ‘morphologically rich language’, *verhnij sloj set* ‘top layer of a neural network’, *predobuchennaja jazykovaja model* ‘pre-trained language model’, *izvlechenie imenovannyh sushhnostej* ‘named entity extraction’, *sovremennij russkij jazyk* ‘modern Russian language’, *znachenie obshhej neopredelennosti* ‘value of the total uncertainty’, *baza znanij wikidata* ‘wikidata

database', *semanticheskij sdvig* 'semantic shift', *nabor dannyh* 'data set', *nejronnaja set'* 'neural network', *rechevoj akt* 'speech act', *analiz tonal'nosti* 'sentiment analysis', *semanticheskij klass* 'semantic class', *jazykovaja model'* 'language model', *imenovannaja sushhnost* 'named entity', *baza dannyh* 'database', *komp'yuternaja lingvistika* 'computational linguistics', *trenirovochnye dannye* 'training data', *grammaticheskij priznak* 'grammatical feature'. As one can see, the algorithm extracts not only terms and keywords, but also free phrases. Nevertheless it produced one of the best results.

4.6 TextRank

TextRank was implemented with *summa* package [22]. This algorithm revealed a large number of unigrams (about 80 percent of the total candidate list) that represent such science terms as *model'* 'model', *zadacha* 'task', *rezul'tat* 'result', *metod* 'method', *issledovanie* 'study', etc. Despite preprocessing and lemmatization, TextRank revealed examples with typos and errors, thus showing the poorest results for both types of evaluation.

4.7 KeyBERT

KeyBERT algorithm was launched into two configurations – the default sentence transformers model (paraphrase-multilingual-MiniLM-L12-v2) and the distilled model (distiluse-base-multilingual-cased-v2). The models were already fine-tuned and could be applied to many languages, including Russian. The third scenario involved CountVectorizer with lists of stop words and pos-patterns.

KeyBERT with vectorizer extracted 4- and 5-grams (for example, *predobuchennaja transformennaja jazykovaja model'* 'pre-trained transformer language model' or *jazykovaja model'* *tipa transformer* 'transformer-based language model') and outperformed other algorithms. Better results compared to other KeyBERT implementations can be explained by more elaborate tuning.

4.8 Author keywords vs Expert evaluation

Expert evaluation shows higher precision for all algorithms compared to author keywords (Table 1 presents the results). This can be explained by the fact that instead of selecting the keywords to be assigned from a pre-set list, authors rely on their own consideration and occasionally may misindicate terminology units. In several cases, the candidate phrase was not labeled as a term by the expert, although it was marked among author terms (for example, the key phrase *dvizhenija golovy* 'head movements' that describes a paper focusing on records for a multimodal corpus).

Low results for comparison with author keywords in a number of cases deal with different lengths of the extracted candidates and author terms (we considered only a complete match). The latter group was mostly represented

Table 1: Precision for comparison with author keywords and expert evaluation.

Algorithm	Precision (author keywords)	Precision (expert)
Joint freq	0.11	0.28
Log-likelihood	0.24	0.35
YAKE	0.06	0.39
RAKE	0.32	0.54
TextRank	0.01	0.17
KeyBERT_paraphrase	0.01	0.12
KeyBERT_distiluse	0.00	0.13
KeyBERT_vectorizer	0.07	0.58

by bigrams or longer word combinations specified by authors themselves and, hence, comparison with this list of terms failed to show high scores. Moreover, author terms may not be found in abstracts, but only in papers themselves, and hence in future we need to evaluate the results across full texts.

5 Conclusion

Automatic keyword extraction cannot replace profound expert evaluation, but it can serve as an initial stage for analysis. Keywords extracted by way of automatic methods can be used to compile thesauri, as well as modern dictionaries, as numerous foreign vocabulary units and borrowings appear in scientific discourse. Lack of labeled data still makes it challenging to perform experiments using supervised machine learning methods. Thus, the collected data can be used for data annotation. At the same time, some models are often pre-trained on more general data (for example, news collections, Wikipedia, web texts), which may impair the quality of the results, making them different from what is desired. For this reason, the next step may suggest training the models on more relevant texts, including compiling a collection of academic papers on linguistics. To this end, the corpus requires significantly more data, considering how demanding are the accuracy requirements.



References

1. Scott, M. PC analysis of key words - and key key words. *System*, 25(2), 233–45 (1997)
2. Dunning, T. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics – Special Issue on Using Large Corpora*, vol. 19, no. 1, 61–74 (1993)
3. Scott, M.: *WordSmith Tools version 8 (64 bit version)* Stroud: Lexical Analysis Software (2022)
4. Kilgarrieff A. Simple maths for keywords, In: *Proceedings of Corpus Linguistics Conference CL2009*, University of Liverpool, UK (2009)
5. Sketch Engine, <http://sketchengine.eu>. Last accessed 6 Nov 2022
6. AntConc, <https://www.laurenceanthony.net>. Last accessed 6 Nov 2022

7. El-Beltagy, S.R., Rafea, A. KP-Miner: A Keyphrase Extraction System for English and Arabic Documents. *Information Systems*, 34(1), 132–144 (2009)
8. Rose, S., Engel, D., Cramer, N., Cowley, W. Automatic Keyword Extraction from Individual Documents. In M. W. Berry & J. Kogan (Eds.), *Text Mining: Theory and Applications*: John Wiley & Sons, 1–20 (2010)
9. Campos R., Mangaravite V., Pasquali A., Jorge A.M., Nunes C., Jatowt A. YAKE! Collection-independent Automatic Keyword Extractor. In: Pasi G., Piwowarski B., Azzopardi L., Hanbury A. (eds). *Advances in Information Retrieval. ECIR 2018* (Grenoble, France. March 26 – 29). *Lecture Notes in Computer Science*, vol 10772, pp. 806–810 (2018)
10. Campos, R., Mangaravite, V., Pasquali, A., Jatowt, A., Jorge, A., Nunes, C. and Jatowt, A. YAKE! Keyword Extraction from Single Documents using Multiple Local Features. In: *Information Sciences Journal*. Elsevier, Vol 509, pp. 257–289 (2020)
11. Mihalcea, R., Tarau, P. TextRank: Bringing Order into Texts. In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, vol. 4. pp. 404–411 (2004)
12. Skrlj, B., Repar, A., Pollak, S. Rakun: Rank-based keyword extraction via unsupervised learning and meta vertex aggregation. In *International Conference on Statistical Language and Speech Processing*, pp. 311–323 (2019)
13. Grootendorst M. KeyBERT: Minimal keyword extraction with BERT, <https://doi.org/10.5281/zenodo.4461265>. Last accessed 6 Nov 2022
14. Popova S., Khodyrev I. Ranking in keyphrase extraction problem: is it suitable to use statistics of words occurrences? In: *Proceedings of the Institute for System Programming of RAS* 26(4), 123–136 (2014)
15. Bruches E., Pauls A., Batura T., Isachenko V. Entity Recognition and Relation Extraction from Scientific and Technical Texts in Russian. In: *Proceedings of the Science and Artificial Intelligence Conference*, p. 41–45 (2020)
16. Nguyen, Q. H., Zaslavskiy, M. Keyphrase Extraction in Russian and English Scientific Articles Using Sentence Embeddings. In *Proceeding of the 28th Conference of Fruct Association*, pp. 334–340 (2021)
17. Sheremet'eva, S., Osminin P. Metody i modeli avtomaticheskogo izvlechenija kljuchevyh slov [Methods and models for automatic keyword extraction]. In *Vestnik of South Ural State University. Series "Linguistics"*, vol. 12, no 1, pp. 76–81 (2015)
18. International Conference on Computational Linguistics and Intellectual Technologies "Dialogue", <https://www.dialog-21.ru/en/>. Last accessed 6 Nov 2022
19. Korobov M.: Morphological Analyzer and Generator for Russian and Ukrainian Languages. In: *Analysis of Images, Social Networks and Texts*, pp. 320–332 (2015).
20. Akhmanova, O.: *Slovar' lingvisticheskikh terminov* [Dictionary of linguistic terminology]. Izdatel'stvo "Sovetskaja enciklopedija", Moscow (1969)
21. Multi_rake package, <https://pypi.org/project/multi\protect\discretionary{\char\hyphenchar\font}{-}{rake}/>. Last accessed 6 Nov 2022
22. Summa package, <https://pypi.org/project/summa/>. Last accessed 6 Nov 2022

Information Extraction from Business Documents

A Case Study

Martin Geletka , Mikuláš Bankovič , Dávid Meluš , Šárka Ščavnická ,
Michal Štefánik , and Petr Sojka 

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
456576@mail.muni.cz

Abstract. Document AI is a relatively new research topic that refers to techniques for automatically reading, understanding, and analyzing business documents. Nowadays, many companies extract data from business documents through manual efforts that are time-consuming and expensive, requiring manual customization or configuration. This paper describes techniques to address these problems, apply them to real-world data, and implement them to an end-to-end solution for automatic information extraction from business documents.

Keywords: OCR, Multi-modal learning, Information extraction, Transformers, Structured Documents

1 Introduction

Information extraction typically consists of two consecutive steps. Firstly text detection and recognition are run to obtain text representation of the input document. Secondly, we extract the information from the received text. In our paper, we present a multi-modal approach to information extraction, which extracts information not only from text alone but integrates all three modalities: text, position, and image, to obtain the best results.

2 OCR frameworks

An essential step at the start of the business document pipeline is finding text blocks and their positions on the page. For scanned documents, OCR (Optical Character Recognition) frameworks are needed.

There are recent OCR frameworks based on deep learning: we describe the models and steps used in these frameworks. We focus on two main steps in OCR pipelines: text detection and text recognition. We discuss the importance of customization and fine-tuning the models included. Specifically, we compare frameworks: Doctr [11], EasyOCR [6], and Tesseract [13] and their ability to be customized and fine-tuned for document understanding in the Czech language.

Text detection models aim to output areas containing text. Most of these models are trained in languages based on Latin script. Therefore, we assume the performance does not suffer without training or fine-tuning the models for the Czech language and is sufficient to perform document understanding. Scene text detection is an active area of research and can be easily re-used in document understanding.

Text recognition models aim to extract text from the bounding boxes generated by text detection models. These models need to be fine-tuned for the specific language vocabulary. We use Differentiable Binarization Net (DBNet) [9] as it is available in both EasyOCR and Doctr¹. We can compare text recognition models in an end-to-end pipeline by unifying text detection architecture. EasyOCR has additionally available CRAFT [3] model.

2.1 Models

This subsection briefly introduces different model architectures we are training or re-using.

DBNet Text detection model, that proposes Differentiable Binarization. The model produces a segmentation map alongside the proposed threshold for binarization. The threshold map solves post-processing and improves metric performance and speed. Model implementations can differ in Convolutional Neural Network (CNN) backbone². The most common is original *vgg-16*, *resnet18*, deeper *resnet50*.

Convolutional Recurrent Neural Network (CRNN)[12] Text recognition model that combines the strengths of image feature extraction of CNN followed by the sequential processing of RNN. CNN's local patch processing ensures that columns from the feature space correspond to column patches in an original image. Locality preserving allows sequential processing in RNN. The most common RNN layers are GRU or LSTM to model long-term dependencies.

MASTER: Multi-Aspect Non-local Network for Scene Text Recognition [10] Text recognition model that introduces Multi-Aspect Global Context Attention (GCAttention) based encoder module and a transformer-based decoder module.

Vision Transformer for Fast and Efficient Scene Text Recognition (ViTSTR) [2] Text recognition model that follows transformers architecture with self-attention mechanism and multi-headed attention. This model emphasizes speed and efficiency in a single-stage encoder step based on vision transformer (ViT) [4].

¹ Implementation of DBNet in EasyOCR and Doctr differ in backbone and weights

² Implementation of CRNN in EasyOCR, Doctr, and Tesseract differ in backbone and weights.

2.2 Tesseract

Tesseract is a well-known OCR framework that is considered to be the open-source baseline. Since version 4.0.0, Tesseract has used the CRNN architecture with the LSTM network for text recognition. Text detection is still performed using multiple steps: component analysis, contour detection, and lines detection. Traditional text detection causes Tesseract to be more prone to preprocessing techniques. In order to get better OCR results, improvement of the quality of the image is needed.

2.3 EasyOCR

EasyOCR framework offers only the CRNN model as a baseline. The CRNN is pre-trained on an English text and it combines Convolutional Neural Networks and Recurrent Neural Networks. EasyOCR's pre-, mid- and post-processing are parametrized and customizable. The parameters for text bounding box merging can change granularity from lines to words, and slope parameters adjust how much rotation is allowed in text bounding boxes. However, training scripts need to be better documented.

2.4 Doctr

Doctr framework is very recent and contains CRNN, MASTER and ViTSTR model architectures. We use this framework for custom training as it contains well documented curated repository with novel architectures.

3 Multi-modal Transformers overview

In this section, we will introduce the multi-modal models, which use additional modalities, such as position and image, to maximize the performance of information extraction tasks. In more detail, we will discuss the Layout Language Model (LayoutLM) Family developed by Microsoft Corporation. We will describe the three generations of the LayoutLM models together with its cross-lingual version LayoutXLM. We will also list related work by other research groups to obtain the whole picture of Multi-modal models.

3.1 LayoutLM family

The first model of the LayoutLM family has introduced already in December 2019. [17] Architecture of this first model was a quite simple extension of the Vanilla Transformer model. Instead of simple WordPiece tokens, this model takes on input also individual positions of the bounding boxes of corresponding tokens. The context-aware embeddings from the Transformer models are then concatenated with the document representation from a pre-trained Vision Neural Network.

The main difference between the first and second versions of the LayoutLM model is that the LayoutLMv2 takes the image representation on the input of Transformer models and therefore is able to train attention between all three modalities at once.

The improvement in the third version of the model was to use a domain-specific model for the Vision Embeddings. The model used a Document Image Transformer pre-trained as Auto-Encoder on IIT-CDIP, a dataset that includes 42 million document images [14].

The main training objective for these models is still a variation of the Mask Language Modelling, which aims to predict masked text tokens based on their position and surrounding text and image context information. The models also use additional pre-training tasks, which can be found in the published paper of all three models. [17,16,5]

All three versions of the models were trained on IIT-CDIP Test Collection [14]. The collection contains 11 million scanned documents. The dataset consists of documents from the state’s lawsuit against the tobacco industry and extracted text provided by the OCR system in the 1990s.

3.2 Multi-lingual models

The LayoutLM family is extended to LayoutXLM (Layout Cross-Lingual Language Model) to address the problem of information extraction from documents from multiple languages. [18] This model architecture and pre-training are inherited from LayoutLMv2 but pre-trained and evaluated on different datasets. The lack of some extended multi-lingual scanned document dataset forced the researchers to crawl the web for digital-born multi-lingual documents. Scrapped were then parsed with a PDF parser and filter from records containing less than 200 words or containing more than one language (identified by language detector from the BlingFire³). The dataset was then enriched by sampling from scanned English documents from IIT-CDIP Test Collection. The final dataset contained more than 30M documents in 53 languages (including Czech and Slovak).

The only other model we found pre-trained on the multi-lingual dataset is LiLT [15]. This model offers the option to divide the text representation and layout into two separate models, which can be pre-trained separately and only fine-tuned together. Therefore in fine-tuning, one can use any pre-trained language model such as XLM-RoBERTa and then only merge it with Layout Transformer and fine-tune them together. For more information about how the textual model is separated, we refer the reader to the original paper [15].

3.3 Other related work

Extensive research exists in multi-modal transformers, but to our knowledge, only LayoutLM and LiLT also offer multi-lingual models. In this section, we provide a short overview of conducted work in this area and for further

³ <https://github.com/microsoft/BlingFire>

information, refer the reader to the original papers. These models also belong to related multi-modal models:

- **FormNet** – model from Google AI Research, which proposes two new mechanisms called Rich Attention and Super-Tokens. Rich Attention leverages the spatial relationships between tokens to calculate a more structurally meaningful attention score and Super-Tokens for each word in a form by embedding representations from their neighboring tokens through graph convolutions. [7]
- **DocFormer** – model from Amazon AI Research Group, which offers another type of multi-modal attention implemented through residual connections and contributes to additional pre-training tasks. [1]
- **SelfDoc** – from Brandeis University and Adobe Research group, which main difference to LayoutLM family models is that it adopts semantically meaningful components (e.g., text block, heading, figure) as the model input instead of WordPiece tokens. [8]

4 Experiments

This section will describe the dataset used for training our models, followed by a description of the individual experiments and a comparison of the trained models.

4.1 Dataset description

We perform a collection of documents, as no Czech documents dataset of a sufficient volume or quality is available. Our collection is performed by querying and automated downloading of the document-format file results from a publicly-available data-sharing platform uloz.to. We query for keywords associated with common categories of office documents, such as “faktura”, “smlouva” or “doklad”. Such-collected, categorized documents are then manually cleaned, resulting in a collection of 6,849 invoice images that we annotate for chosen entity types.

We obtained languages of crawled documents by applying publicly available language detection tool⁴ on the output of the Tesseract OCR engine.

In Figure 3, we can see that the final dataset contains documents mainly in Czech, Slovak, and Polish but also small amounts of English and Slovenian invoices.

The annotation process consists of (i) selecting a bounding box (BBox) that *separates* a position of the entity within the document visual and (ii) assigning a category to such BBox out of a predefined set of entity labels⁵. In Figure 1, we

⁴ Tool is available at <https://github.com/Mimino666/langdetect>

⁵ The entity types are chosen to allow an automated payment of the detected invoice based on the extracted information.

can see an example of an invoice with annotated entities with corresponding bounding boxes.

The annotation process yields a total of 39,670 entity annotations, ranging from 10 annotations for the rarest category (specific symbol) to 8,359 annotations for the most common category (total amount).

4.2 Born-digital dataset

By the same procedure, we receive a collection of 788 pdf documents consisting of born-digital invoices and scanned invoices. Firstly, we clean the dataset by removing scanned documents. As a criterion for scan identification, we use the average size of the page, dimensions of images, and characters. More precisely, we reject documents with an average page size greater than 500,000B or documents containing images with dimensions larger than the dimensions of the pdf. Further, we inspect characters, and documents containing unknown characters are removed. Through this process, we obtain 485 documents that are processed and used for fine-tuning Doctr and EasyOCR models.

We use pdfminer program to extract words (labels) and the corresponding images, resulting in a dataset of 687,241 samples. We train OCR models on this dataset with a train-validation-test split on unique words (60/20/20).

4.3 Results OCR


In Table 1, we compare pre-trained Tesseract, EasyOCR, and Doctr CRNN models with our trained models Doctr MASTER and Doctr ViTSTR. These models are not available pre-trained, and our training of the Doctr CRNN model was unsuccessful due to an error in the library. We compared the exact match, partial match, and elapsed time. The exact match is a 1 – word error rate. A partial match is 1 if the ground truth starts with the full prediction; otherwise, 0.

Doctr MASTER performed the best from our tested models with 2% word error rate. However, it is a magnitude slower than Tesseract. Doctr ViTSTR is the faster-trained model; however, its performance is insufficient for commercial use. Pre-trained EasyOCR model is faster and has better performance than Doctr ViTSTR.

4.4 Results

In this section, we will compare the performance of text-only models with the same-sized LayoutLM models. We also compared models with two different pretraining datasets, i.e. pretrained only on English data and pretrained on multiple languages, including Czech and Slovak.

As a representative of the text-based model pretrained on English, we chose RoBERTa Large model and pretrained on the multilingual dataset BERT Vase Multilingual Cased, XLM RoBERTa Base, and XLM RoBERTa Large. From the

 VIDOX STAVEBNÍ	Variabilní symbol (uvádějte při platbě): 300008616																						
	Strana č. 1																						
Faktura - daňový doklad č.: 300008616																							
Dodavatel: VIDOX s.r.o. U Poráků 511, Horní Brána 381 01 Český Krumlov Česká republika IČ: 25160168 DIČ: CZ25160168 Stavební divize: Vodárenská 1091/II, 379 01 Třeboň <small>Dodavatel je registrován pod spíšeovou značkou oddíl C, vložka 6019 za dne 24.03.1997 u Krajského soudu v Českých Budějovicích.</small> Úhrada: Na bankovní účet Banka: Komerční banka a.s. Český Číslo účtu: 4255580287/0100 IBAN: CZ5401000000004255580287 SWIFT: KOMBCZPPXXX	Odběratel: Zákaznické číslo: 107636 Husitské muzeum v Táboře nám. Mikuláše z Husi 44/5 390 01 Tábor IČ: 00072486 DIČ: CZ00072486	Husitské muzeum v Táboře datum: 10-08-2016 č. j.: HH-CP/664/2016 listy: 1 přílohy: 4 zpracoval: [signature] splatnost:																					
Středisko: 300 Akce: Datum vystavení dokladu: 5.8.2016 Datum zdanitelného plnění: 31.7.2016 Místo plnění: CZ	Datum splatnosti: 4.9.2016																						
<table border="1"> <thead> <tr> <th>Předmět zdanitelného plnění</th> <th>Množství j.</th> <th>Cena za jedn. v CZK bez</th> <th>Cena celkem bez DPH</th> <th>Sazba DPH</th> <th>Částka DPH</th> <th>Cena celkem s DPH</th> </tr> </thead> <tbody> <tr> <td colspan="7"> Fakturuje se Vám provedené stavební práce na stavební zakázce "Stavební úpravy a přístavba č. p. 2073 Tábor" dle uzavřené ŠOP č. objednatele 5/2016/PO za období 1.7.2016 - 31.7.2016 a zjišťovacího protokolu č. 5, který tvoří nedílnou součást této faktury částka ve výši: </td> </tr> <tr> <td></td> <td></td> <td>1 455 507,17</td> <td>0,00</td> <td>0,00</td> <td>1 455 507,17</td> <td></td> </tr> </tbody> </table>	Předmět zdanitelného plnění	Množství j.	Cena za jedn. v CZK bez	Cena celkem bez DPH	Sazba DPH	Částka DPH	Cena celkem s DPH	Fakturuje se Vám provedené stavební práce na stavební zakázce "Stavební úpravy a přístavba č. p. 2073 Tábor" dle uzavřené ŠOP č. objednatele 5/2016/PO za období 1.7.2016 - 31.7.2016 a zjišťovacího protokolu č. 5, který tvoří nedílnou součást této faktury částka ve výši:									1 455 507,17	0,00	0,00	1 455 507,17			
Předmět zdanitelného plnění	Množství j.	Cena za jedn. v CZK bez	Cena celkem bez DPH	Sazba DPH	Částka DPH	Cena celkem s DPH																	
Fakturuje se Vám provedené stavební práce na stavební zakázce "Stavební úpravy a přístavba č. p. 2073 Tábor" dle uzavřené ŠOP č. objednatele 5/2016/PO za období 1.7.2016 - 31.7.2016 a zjišťovacího protokolu č. 5, který tvoří nedílnou součást této faktury částka ve výši:																							
		1 455 507,17	0,00	0,00	1 455 507,17																		
Všechny zakázky - TENDERMARKET pod ev. č. T004/15V/0004892																							
Upozornění: 1) Daň odvede zákazník.																							

Částky v CZK			
	Bez DPH	DPH	Celkem
0 %	1 455 507,17	0,00	1 455 507,17
Celkem	1 455 507,17	0,00	1 455 507,17
Zaokrouhlení			0,00
Na zálohách zapláceno			0,00
Částka k úhradě			1 455 507,17

Základem pro výpočet daně je částka "Bez DPH".

Vystavil(a): Kateřina Hloušková

Převzal(a), dne: [signature] převzal a souhlasí:



Vytvářeno v systému ADRAG3

Telefon: +420384721357

Fax: +420384721357

E-mail: katerina.hlouskova@vidox.cz

Mobilní telefon:

WWW: www.vidox.cz

Fig. 1: Example of annotated invoice

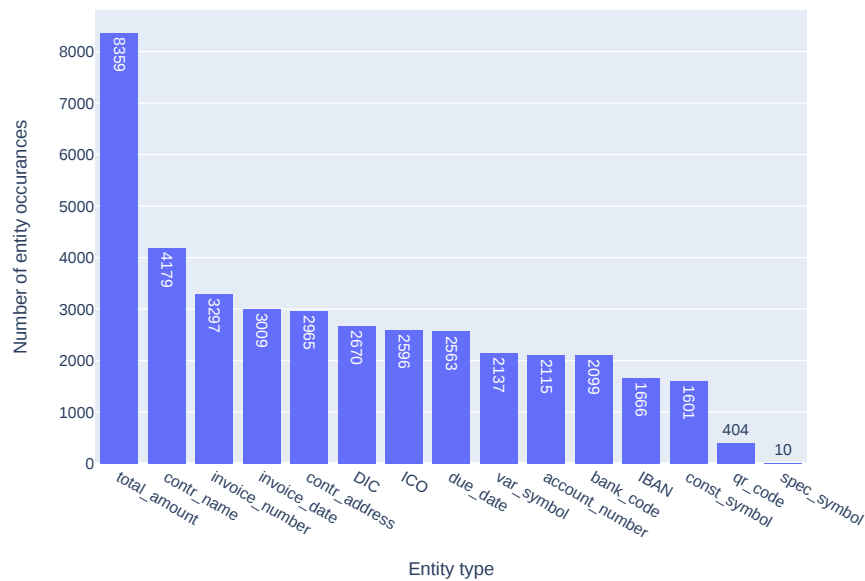


Fig. 2: Number of individual entity types across the whole dataset.

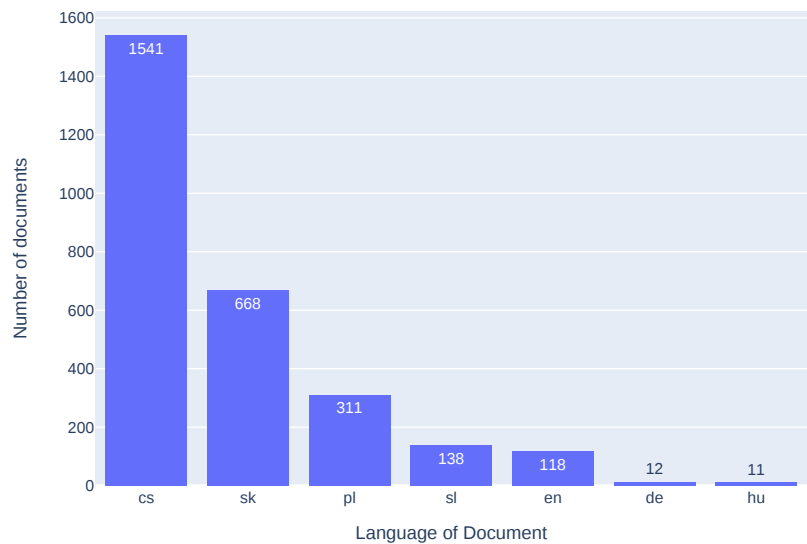


Fig. 3: Number of documents per language identified by language detection tool (visualizing only languages with more than 5 documents).

Table 1: Performance comparison of text recognition models on born-digital dataset

	Exact	Partial	FPS
Tesseract v5	0.90	0.90	3.35
EasyOCR CRNN	0.83	0.84	34.14
Doctr CRNN	0.89	0.89	27.36
Doctr MASTER	0.98	0.99	0.46
Doctr ViTSTR	0.75	0.83	18.84

LayoutLM family, we fine-tuned two models pretrained on English scanned documents: LayoutLMv2 Base and LayoutLMv2 Large, and pretrained on a multilingual dataset: LayoutXLM Base.

We used the Tesseract OCR engine for all models to extract text information from the annotated scanned dataset.

In Table 2, we can see that pretrained multi-modal achieved higher scores than their equal-sized tex-only-based models. Furthermore, we see that both text-based and multi-modal models improved more by increasing the model size than by including multilingual pretraining since the best-performing text-only model is XLM RoBERTa Large, which is the strongest text-only model, and the overall best-performing model is LayoutLMv2 Large.

In Figure 4, we can see the confusion matrices of two best-performing multimodal models: LayoutXLM base and LayoutLMv2 large.

Table 2: Performance comparison of Text-based and LayoutLM models on separated evaluation datasets.

	F1-score	Precision	Recall
BERT Base Multilingual Cased	66.74	66.75	66.73
XLM RoBERTa Base	72.80	72.61	73.00
RoBERTa Large	78.25	77.52	79.00
XLM RoBERTa Large	79.36	80.30	78.44
LayoutLM v2 Base	77.83	75.99	79.76
LayoutLM v2 Large	83.06	82.38	83.75
LayoutXLM Base	79.40	78.75	80.06

5 Discussion and Future Work

In this paper, we presented end to end solution for information extraction in business documents. We offered solutions for both OCR and information extraction by text-only and multi-modal Transformers.

DIC	92.84	0	1.25	15.06	0	0	0	0	0	0	0	0	0	0	0	0	0.83
IBAN	0	97.2	0	2.23	0.27	0	0	0	0	0	0	0	0	0	0	0	0
ICD	1.53	0	84	8.61	0.3	0	0.61	0.61	1.84	0	0	0.3	0	0	0	0	2.15
O	0.1	0.05	0.06	98.27	0.03	0	0.01	0.5	0.6	0.07	0.1	0.04	0.03	0	0.24	0.02	
account_number	4.22	0	0	22.53	72.53	0.7	0	0	0	0	0	0	0	0	0	0	0
bank_code	0	0	0	15.78	7.36	96.84	0	0	0	0	0	0	0	0	0	0	0
const_symbol	0	0	0	12.96	0	0	97.03	0	0	0	0	0	0	0	0	0	0
const_address	0	0	0.05	10.19	0	0	0	88.04	1.69	0	0	0	0	0	0	0	0
const_name	0	0	0	14.06	0	0	0	1.94	93.98	0	0	0	0	0	0	0	0
due_date	0	0	0	11.34	0	0	0	0	0	97.23	0.92	0	0	0	0	0	0
invoice_date	0	0	0.24	15.13	0	0	0	0.24	0.49	2.48	81.38	0	0	0	0	0	0
invoice_number	0	0	0	12.2	0	0	0.33	0	0	0	0	92.73	0	0	0	4.74	0
qr_code	0	0	0	7.2	0	0	0	0	0	4	0	0	24	0	0	0	0
spec_symbol	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0
total_amount	0	0	0.09	11.63	0	0.09	0	0	0	0	0	0	0	0	0	88.17	0
var_symbol	0	0.76	0	5.38	0	0	0	0	0	0	0	3.84	0	0	0	96	0
Actual Category	DIC	IBAN	ICD	O	account_no.	bank_code	const_sym.	const_addr.	const_name	due_date	invoice_date	invoice_no.	qr_code	spec_symbol	total_amount	var_symbol	

DIC	85.65	0.43	2.6	10.43	0	0	0	0	0	0	0	0	0	0	0	0	0.86
IBAN	0	97.37	0	2.62	0	0	0	0	0	0	0	0	0	0	0	0	0
ICD	1.47	0.73	83.03	9.55	0	0	0	4.77	0.36	0	0	0	0	0	0	0	0
O	0.07	0.03	0.04	98.36	0.03	0	0.02	0.34	0.54	0.04	0.12	0.08	0	0	0.25	0.01	
account_number	0.4	0.4	0	21.37	72.28	0.8	0	0	0	0	0	0	0	0	0	0	0
bank_code	0	0	0	29.21	6.74	94.04	0	0	0	0	0	0	0	0	0	0	0
const_symbol	0	0	0	3.77	0	0	96.22	0	0	0	0	0	0	0	0	0	0
const_address	0	0	0.18	10.8	0	0	0	87.42	1.58	0	0	0	0	0	0	0	0
const_name	0	0	0	14.49	0	0	0	2.14	83.35	0	0	0	0	0	0	0	0
due_date	0	0	0	9.03	0	0	0	0	0	90.32	0.64	0	0	0	0	0	0
invoice_date	0.27	0	0.27	11.5	0	0	0.27	0.54	0.27	86.57	0	0	0	0	0	0.27	
invoice_number	0	0	0	10.8	0	0	0	0	0	0	88.15	0	0	0	0	1.04	
qr_code	0	0	0	65.71	0	0	0	0	0	0	0	0	14.28	0	0	0	0
spec_symbol	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0
total_amount	0	0	0	13.52	0	0.14	0	0	0	0	0	0	0	0	0	86.32	0
var_symbol	0	0	0	2.34	0	0	0	0	0	0	0	1.56	0	0	0	96.09	0
Actual Category	DIC	IBAN	ICD	O	account_no.	bank_code	const_sym.	const_addr.	const_name	due_date	invoice_date	invoice_no.	qr_code	spec_symbol	total_amount	var_symbol	

Fig. 4: Confusion matrices of finetuned LayoutXLM base (left) and LayoutLMv2 large (right) models.

In the OCR sections of the paper, we proved that EasyOCR and Doctr framework are sufficient for document understanding. However, to create novel and well-behaving products, custom training is necessary. Both have the common intersection of models; however, models are not compatible with each other and prevent any sensible comparison. Training scripts for EasyOCR need to be more documented. Doctr framework lacks parametrization. The training can be improved by hyper-parameter search, as Transformers and CRNN have a different sensibility to learning rate and the number of epochs. Our paper does not evaluate the text detection models, which can cause a bottleneck in the commercial use of the system.

In the NER sections of the paper, we offered an overview of available multi-modal Transformer models and, more in detail described the family of the LayoutLM models. In the Section 4, the LayoutLM model with equal size achieved significantly better results than their equally-sized text-only counterparts. We also show that on presented dataset size of the model improved results by a higher margin than introducing multilingual pretraining. This behavior can be explained by the types of extracting entities, mainly composed of information written in digits (number codes, dates, times, sum), which are language-independent.

In future work, we propose experimenting with the LayoutLMv3 model as we revealed that models pretrained only on English could outperform multilingual-based models. Furthermore, we plan to research the impact of the used OCR engine on the performance of the NER model.

Acknowledgements We acknowledge the support of grant Intelligent Back Office, project number CZ.01.1.02/0.0/0.0/21_374/0026711.

References

1. Appalaraju, S., Jasani, B., Kota, B.U., Xie, Y., Manmatha, R.: DocFormer: End-to-End Transformer for Document Understanding. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 973–983 (2021). <https://doi.org/10.1109/ICCV48922.2021.00103>
2. Atienza, R.: Vision transformer for fast and efficient scene text recognition. In: International Conference on Document Analysis and Recognition. pp. 319–334. Springer (2021)
3. Baek, Y., Lee, B., Han, D., Yun, S., Lee, H.: Character Region Awareness for Text Detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (Jun 2019)
4. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. CORR (2020), <https://arxiv.org/abs/2010.11929v2>
5. Huang, Y., Lv, T., Cui, L., Lu, Y., Wei, F.: LayoutLMv3: Pre-Training for Document AI with Unified Text and Image Masking. In: Proceedings of the 30th ACM International Conference on Multimedia. pp. 4083–4091. MM '22, Association for Computing Machinery, New York, NY, USA (2022). <https://doi.org/10.1145/3503161.3548112>
6. Jaied AI: Easyocr. <https://github.com/JaiedAI/EasyOCR> (2020)
7. Lee, C.Y., Li, C.L., Dozat, T., Perot, V., Su, G., Hua, N., Ainslie, J., Wang, R., Fujii, Y., Pfister, T.: FormNet: Structural encoding beyond sequential modeling in form document information extraction. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 3735–3754. ACL, Dublin, Ireland (May 2022). <https://doi.org/10.18653/v1/2022.acl-long.260>, <https://aclanthology.org/2022.acl-long.260>
8. Li, P., Gu, J., Kuen, J., Morariu, V.I., Zhao, H., Jain, R., Manjunatha, V., Liu, H.: SelfDoc: Self-Supervised Document Representation Learning (2021). <https://doi.org/10.48550/ARXIV.2106.03331>
9. Liao, M., Zou, Z., Wan, Z., Yao, C., Bai, X.: Real-time scene text detection with differentiable binarization and adaptive scale fusion. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2022). <https://doi.org/10.1109/TPAMI.2022.3155612>
10. Lu, N., Yu, W., Qi, X., Chen, Y., Gong, P., Xiao, R., Bai, X.: Master: Multi-aspect non-local network for scene text recognition. Pattern Recognition **117**, 107980 (2021)
11. Mindee: docTR: Document Text Recognition. <https://github.com/mindee/doctr> (2021)
12. Shi, B., Bai, X., Yao, C.: An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(11), 2298–2304 (2017). <https://doi.org/10.1109/TPAMI.2016.2646371>
13. Smith, R.: An overview of the Tesseract OCR engine. In: Ninth international conference on document analysis and recognition (ICDAR 2007). vol. 2, pp. 629–633. IEEE (2007)
14. Soboroff, I.: Complex Document Information Processing (CDIP) dataset (2022). <https://doi.org/10.18434/mds2-2531>
15. Wang, J., Jin, L., Ding, K.: LiLT: A Simple yet Effective Language-Independent Layout Transformer for Structured Document Understanding (2022). <https://doi.org/10.48550/ARXIV.2202.13669>

16. Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., Lu, Y., Florencio, D., Zhang, C., Che, W., Zhang, M., Zhou, L.: LayoutLMv2: Multi-modal Pre-training for Visually-rich Document Understanding. In: Proceedings of the 59th Annual Meeting of the ACL and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 2579–2591. ACL, Online (Aug 2021). <https://doi.org/10.18653/v1/2021.acl-long.201>, <https://aclanthology.org/2021.acl-long.201>
17. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., Zhou, M.: LayoutLM: Pre-Training of Text and Layout for Document Image Understanding. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1192–1200. KDD '20, Association for Computing Machinery, New York, NY, USA (2020). <https://doi.org/10.1145/3394486.3403172>
18. Xu, Y., Lv, T., Cui, L., Wang, G., Lu, Y., Florêncio, D., Zhang, C., Wei, F.: LayoutXLM: Multimodal Pre-training for Multilingual Visually-rich Document Understanding. CoRR **abs/2104.08836** (2021), <https://arxiv.org/abs/2104.08836>

Keyword Extraction for Automatic Evaluation of Machine Translation

Lívia Kelebercová and František Forgáč

Department of Informatics, Faculty of Natural Sciences and Informatics,
Constantine the Philosopher University in Nitra,
Trieda Andreja Hlinku 1, 94974 Nitra,
livia.kelebercova@ukf.sk

Abstract. Evaluation plays a key role in the field of machine translation. In general, the evaluation of machine translation can be divided into two types, manual (human) and automatic (machine). Professional human translators can understand and evaluate the text with the best results in terms of measuring quality and analyzing errors but on the other hand, this approach brings a number of disadvantages, including high time consumption, the subjectivity of the translator, and the financial costs associated with hiring professional translators. Automatic evaluation approaches are usually based on the correlation between the sentences or n-grams from human translation and machine translation. The aim of this paper is to capture the semantics of human translation from the English language to the Slovak language and the same text translated by ETransL and DeepL translating engines by extracting the keywords which represent the main phrases from documents to determine how much the machine translations differ from the reference human translation. Based on our results the translations are equal from a semantic point of view and the end user should understand the text translated by ETransL and DeepL equally as human translation.

Keywords: Keyword Extraction, Machine Translation, Evaluation of Machine Translation

1 Introduction

Machine translation evaluation is necessary to discover how closely the neural translation language model relates to the reference domain. This process is essential for determining the effectiveness of an existing model and estimating the amount of post-processing the model needs to fulfill the expectations of end-users.

Evaluation approaches can be basically divided into manual approaches and automatic approaches. Manual approaches use professional human translators to evaluate key metrics such as adequacy and fluency scores. The main problem with the manual approach is that evaluation is based on subjective human judgment and this process is time-consuming [1].

1.1 Related Work

To solve the problems with human evaluation researchers developed automatic approaches. These approaches are trying to evaluate how close the machine translation is to one or more human references by using metrics as BLEU [2], NIST [4], or METEOR [5]. These metrics usually score individual segments which are usually sentences and the main concept behind them is that the closer a machine translation is to a reference translation, the better it is [1,2].

Despite the fact that BLEU [2] is the most common metric to evaluate the quality of machine translation, the reliability is questionable. Based on Babych's research, the main problem is that blue and many other commonly used metrics measure lexical identity at the surface level but they are insensitive to linguistic variations [5,6]. Some metrics such as METEOR try to solve this problem by including semantic tools like WordNet lexical database to reduce the dependence on exact matches of words in sentences. WordNet-based approaches also have their disadvantages and may not be able to fully describe word similarity between MT out-puts and references [6]. An interesting way was presented by Mirsarraf and Deghani [7] who, inspired by Lo's [8] research, and like him, proposed a depend-ency-inspired semantic evaluation methodology to quantify how well the underlying meaning of the source is maintained in the translated output using dependency analysis concepts in SRL. Several researchers have attempted to include semantics in machine translation evaluation but neither of them was trying to include keyword extraction in evaluation metrics.

1.2 Proposed Method

In this paper, we focused on finding similarities between machine translation and human translation in terms of text observation rather than in the context of appropriateness and adequacy for each word/phrase/sentence. To capture the meaning of the text we used keyword extraction. Keyword extraction is used to identify the most important phrases from the document [4,5]. From a linguistic point of view, if the machine translation is equal to human translation, then applying a keyword extraction algorithm should give the same keywords for each text. If we don't get the same keywords, it means that the translations are probably different. The Slovak language is one of the inflected languages, and therefore the extracted keywords may differ in endings, which means that there are machine translation errors, but only from the grammar point of view. For this reason, we used a higher level of granularity and determined the base of the word and the root of the word. According to the lemma, we were able to determine the part of speech of the root of the word (stem). There are two reasons why we determined stem. The first reason is that if the stem was the same for the extracted keywords, it could have resulted in the change of part of speech. The second case represents that the meaning is preserved, but the resulting form is incorrect and there is a fluency error. Formally speaking, the neural machine translation model mistakenly transfers the abstract representation of the source word to the abstract representation

of the target language and a shift in meaning occurs, or the language model correctly transforms the abstract representation of the word into the target language, but in the target language, it erroneously creates the external form of the given word.

The rest of our paper is organized as follows. In section 2 we describe the keyword extraction process and data preparation for the next step. Section 3 presents the evaluation process and results and in Section 4 we evaluate the proposed method.

2 Methods

The keyword extraction subsection is intended to introduce you to the function of the chosen approach for extracting key phrases. The subsection data preparation serves to describe the further processing of data for the purpose of subsequent evaluation

2.1 Keyword Extraction

In the case of estimating the quality of machine translation, we used text translated from English to the Slovak language by a human as a reference translation and the same text translated by ETransL [9] and DeepL [10] machine translators. To improve the process of Keyword Extraction it was necessary to preprocess our text data. At first, we converted our data to lowercase. Then we removed tags and special characters. The last step was to identify stop words. For this purpose, we used a library that contains 418 Slovak stop words.

Keyword Extraction is a summarization technique, which uses statistical information from the text to identify the most important phrases [12,15]. In this case, we used a Rapid Keyword Extraction Algorithm (RAKE) [13].

The main concept behind the RAKE algorithm is that keywords are often consisting of multiple words without any interpunction or stop words. The algorithm is based on collocation and co-occurrence, which means the goal is to find words that are frequently occurring together in desired n-gram range [13].

For implementation, we used a rake python package [14]. We created a function that accepts a list of stop words, preprocessed text, and n-gram range as parameters to initialize the RAKE algorithm. The first step of the algorithm is to split the text into words and place them in the word degree matrix. We can imagine this matrix as an Excel table, where every word is placed in separate cells horizontally and vertically. Then each word is assigned a score presenting how frequently a given word co-occurs with another word [13].

The next step is to calculate the degree of the word in the matrix, which presents the sum of the number of co-occurrences divided by the frequency (how many times a word occurs in the corpus). The final score for keywords in desired n-gram range is then calculated as a sum of degrees of words of its words [13]. We called this function on each translation to obtain the top 100 keywords consisting of two words and the top 100 keywords consisting of three words for each text separately.

2.2 Data preparation

From the Keyword Extraction process, we obtained 100 key phrases consisting of two words and 100 key phrases consisting of three words for human translation, ETransL and Deepl translation. We took all 600 keywords and split them into single words and removed duplicates. We put them in a python data frame column labeled as the word. First, we needed to capture the morphological properties of the words. We used the Stanza [15] library for lemmatization, and we adapted the code of Czech stemmer [16] to obtain the stems in the Slovak language. Lemmas and the stems were then manually controlled and corrected. We placed our lemmas and bases of words in the first column under the original words from keywords and we created a column labeled as level describing whether the word is the original form of the word, lemma, or the stem. The next variable is presented in the tag column. To obtain tags we used MorphoDiTa [17] tagger with a Slovak model.

We calculated the number of times each word occurred in extracted bigrams and trigrams from each translation. These frequencies are presented in columns count in ETransL (2, 2), count in Deepl (2, 2), count in human (2, 2), count in ETransL (3, 3), count in Deepl (3, 3), count in human (3, 3). We also calculated the count of occurrence of each word in the whole translation, so we created columns with labels count in ETransL (full text), count in Deepl (full text), and count in human (full text).

Another value we wanted to capture is the length of the word which is presented in the number of characters in the word column. In the end, we calculated term frequency – the frequency of a word in text divided by the number of words in a document. These values are presented in columns TF etransl, TF deepl, and TF human. We exported our python data frame to an Excel sheet. The First three rows from our sheet are visible on Figure 1.

words	level	tag	Count in Deepl (2,2)	Count in human (2,2)	count in ETransL (3,3)	count in Deepl (3,3)	count in human (3,3)	count in ETransL (full text)	count in Deepl (full text)	count in human (full text)	number of characters	TF etransl	TF deepl	TF human
abdikácia	word	NN	0	0	1	1	0	1	1	0	9	0,00033	0,0003	0
akákoľvek	word	PZ	0	0	0	0	1	0	0	1	9	0	0	0,0003
alexandra	word	NN	0	0	3	1	1	12	10	7	9	0,00394	0,0032	0,0022

Fig. 1: First three row from Excel sheet

3 Results

Figure 2 shows the point and interval estimation of the mean in the number of characters. It is logical that the number of characters decreases with the level of granularity, and we have to compare frequencies for each granularity separately.

Level of Granularity	N	Number of characters Mean	Number of characters Std.Dev.	Number of characters Std.Err	Number of characters -95.00%	Number of characters +95.00%
word	952	7,400	2,419	0,078	7,246	7,554
lema	752	7,109	2,339	0,085	6,942	7,276
stem	730	6,163	2,173	0,080	6,005	6,321

Fig. 2: Point and interval estimation of the mean in the number of characters

To verify the effectiveness of the proposed models, we used modified tests for repeated measurements (Greenhouse-Geisser adjustment), due to the violation of the sphericity condition of the covariance matrix. If the condition of sphericity of the covariance matrix is not met, the size of the error of the first type increases. Epsilon represents the degree of violation of the sphericity condition. Epsilon equal to one represents the fulfillment of the condition. Conversely, the smaller it is, the more the condition of sphericity is violated. In our case, the Epsilon values were significantly less than one (word: G-G Epsilon = 0,24, $p < 0,001$; lemma: G-G Epsilon = 0,175, $p < 0,001$; stem: G-G Epsilon = 0,190, $p < 0,001$). Null hypotheses with 99,9% reliability (at the 0,001 significance level) are rejected, which claim that there is no statistically significant difference in word/lemma/stem frequencies in bigrams, trigrams, and whole texts of the ETransL, Deepl, and human translations. Hypotheses were tested at individual levels (word/lemma/stem).

The surrounding of the word is important, it makes a difference whether we compare the frequency of occurrence of the keyword in bigrams, trigrams, or within the entire text. The given keyword/lemma/stem was found in different frequencies, either in the wider area or in the shorter. That's why we compared the translations multiple times to find out between which of them there are statistically significant differences and vice versa between which are not.

From the point of view of multiple comparisons, we identified three homogeneous groups (**** - $p > 0,05$) in frequencies at the word level and two at the lemma/stem level. Naturally, a statistically significant difference was demonstrated between the frequencies in whole texts and in bigrams/trigrams. If we look at the frequency separately for bigrams, trigrams, and whole texts, there are no statistically significant differences between the translations machine and human translations except for frequency at the word level, where a statistically significant difference between ETransL and human was demonstrated ($p < 0,05$).

Although the averages are low, as some keywords occurred just once, as the context of the phrase expands, the frequency of occurrence increases relative to the word/lemma/stem.

Figure 3 shows that there is no statistically significant difference between human translation and machine translation if we only consider bigrams and trigrams. If we consider the whole text, there is a statistically significant difference

in the frequency of occurrence of the keyword between the whole text and the phrases, regardless of whether it is human or machine translation.

level=word	Mean	1	2	3
Count in etranslate (2,2)	0,236	****		
Count in deepl (2,2)	0,239	****		
Count in human (2,2)	0,256	****		
Count in etranslate (3,3)	0,362	****		
Count in deepl (3,3)	0,366	****		
Count in human (3,3)	0,373	****		
Count in etransl (full text)	1,417		****	
Count in deepl (full text)	1,470		****	****
Count in human (full text)	1,570			****

Fig. 3: Frequencies on word level

Interestingly if we take a closer look at the whole text, there is no statistically significant difference between the human translation and DeepL in the frequency of keywords, from which we can conclude that both translations reach the same level in understanding the text (they captured the same keywords, i.e. meaning and even their form, i.e. fluidity). A statistically significant difference was demonstrated between ETransL and human translation, and here we can discuss whether the inaccuracy occurred only in the form of the word (i.e. in the ending, fluency) or also in the lemma or at the root of the word (in meaning, i.e. accuracy).

level=lema	Mean	1	2
Count in human (2,2)	0,160	****	
Count in deepl (2,2)	0,174	****	
Count in etranslate (2,2)	0,178	****	
Count in deepl (3,3)	0,214	****	
Count in etranslate (3,3)	0,231	****	
Count in human (3,3)	0,246	****	
Count in etransl (full text)	1,199		****
Count in deepl (full text)	1,217		****
Count in human (full text)	1,290		****

Fig. 4: Frequencies on lemma level

If we look at Figure 4, we can see that due to the lemma, there is no longer a statistically significant difference, probably a grammatical problem, not a semantic one, that occurs with ETransL. This is also confirmed by Figure 5.

level=stem	Mean	1	2
Count in etranslate (2,2)	0,318	****	
Count in human (2,2)	0,334	****	
Count in deepl (2,2)	0,336	****	
Count in etranslate (3,3)	0,474	****	
Count in deepl (3,3)	0,492	****	
Count in human (3,3)	0,493	****	
Count in etransl (full text)	2,323		****
Count in deepl (full text)	2,349		****
Count in human (full text)	2,460		****

Fig. 5: Frequencies on stem level

4 Conclusion

Our research indicates that due to the extracted keywords, or their frequency throughout the text is no difference between machine translations and human translation. Considering the quality of the translation from a semantic point of view, the translations are equal and the end user/reader should understand the text equally and receive the same information. However, due to the form of the given information, a difference between human translation and ETransL was demonstrated in favor of human translation, i.e. keyword was more common in human translation than in ETransL.

The proposed method of determining translation quality differs from existing approaches in the sense that we were not concerned with determining the quality of machine translation in the context of fluency and adequacy for each word/phrase/sentence, but rather with determining the similarity of machine translation and human translation from the point of view of text observation. In our case, it was journalistic texts whose function is to inform the reader and to get an answer to the questions Who? What? When? Where? and how? Basically, we were concerned with the applicability of machine translation in the given context or domain. Through our research, we have shown that DeepL is usable and very similar to human translation in keywords, starting from phrases first, then words, lemmas, and stems.

Acknowledgments This work was supported by the Slovak Research and Development Agency under contract No. APVV-18- 0473.

References

1. Sepesy Maučec, M., Donaj, G.: Machine translation and the evaluation of its quality. *Recent Trends in Computational Intelligence*. (2020).
2. Papineni, K., Roukos, S., Ward, T., Zhu, W.-J.: Bleu. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*. (2001).
3. Doddington, G.: Automatic evaluation of machine translation quality using N-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research -*. (2002).
4. Denkowski, M., Lavie, A.: Meteor Universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. (2014).
5. Babych, B., Hartley, A.: Sensitivity of Automated MT Evaluation Metrics on Higher Quality MT Output: BLEU vs Task-Based Evaluation Methods. In *The Sixth International Language Resources and Evaluation (LREC'08)* (2008).
6. Wong, B.: Semantic Evaluation of Machine Translation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA), Valletta, Malta (2010)
7. Mirsarraf, M.R., Dehghani, N.: A dependency-inspired semantic evaluation of machine translation systems. *Lecture Notes in Computer Science*. 71–74 (2013).
8. Lo, C., Wu, D.: MEANT: An inexpensive, high-accuracy, semi-automatic metric for evaluating translation utility via semantic frames. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 220–229 (2011)
9. Chribonn: ETRANSL, <https://www.alanbonnici.com/2022/03/etransl.html>.
10. Kutylowski, J.: Deepl, <https://www.deepl.com/en/publisher/> (2017)
11. Kelebercová, L., Munk, M.: Analysis of the Popularity Rate of Extracted Keywords From True and Fake News Related to Covid-19. In *International Scientific Conference on Distance Learning in Applied Informatics*. pp. 384–392. Wolters Kluwer, Prague, Czech Republic (2022).
12. Kelebercová, L., Munk, M.: Search queries related to COVID-19 based on keyword ex-traction. In *Procedia Computer Science*. pp. 2618–2627 (2022).
13. Rose, S., Engel, D., Cramer, N., Cowley, W.: Automatic keyword extraction from individual documents. *Text Mining*. 1–20 (2010).
14. Sharma, V.: Rake nltk, <https://csurfer.github.io/rake-nltk/>, (2021).
15. Qi, P., Zhang, Y., Bolton, J., Manning, C.: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. pp 101–108 (2020)
16. Gomes, L.: Czech Stemmer https://research.variancia.com/czech_stemmer/ (2010)
17. Straková J., Straka, M., Hajič, J. : Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. pp. 13-18. Baltimore, Maryland (2014)

Using NVH as a Backbone Format in the Lexonomy Dictionary Editor

Miloš Jakubíček, Vojtěch Kovář, Michal Měchura, and Adam Rambousek

Natural Language Processing Centre
Faculty of Informatics, Masaryk University, Brno, Czechia
{jak,xkovar3,xrambous}@fi.muni.cz

Lexical Computing
Brno, Czechia
{milos.jakubicek,vojtech.kovar,adam.rambousek}@sketchengine.eu

Abstract. In this paper we present an ongoing development in the Lexonomy dictionary editor consisting of replacing the XML backbone of the editor with an NVH-based one. We describe the core properties of the recently introduced NVH format, implications for using it in Lexonomy as well as a self-contained Python implementation in the form of one script (`nvh.py`) that can be used for several standard processing operations such as parsing, serialization, search or schema validation. We also outline some planned future development related to the usage of NVH in Lexonomy.

Keywords: NVH, XML, Lexonomy, dictionary editor

1 Introduction

This paper focuses on the development of Lexonomy, a lightweight dictionary editing system [1,2]. Lexonomy is a web-based tool which allows users to upload, edit and publish their dictionaries. It is tightly bound to the Sketch Engine corpus management system [3]: users can easily import corpus content from Sketch Engine into Lexonomy, either manually (pull corpus examples, collocations or thesaurus items) or automatically (by using the OneClick Dictionary approach [4]) or in a post-editing fashion [5]) where the dictionary is initially drafted fully automatically and post-edited in isolated steps inside of Lexonomy.

From the beginning, dictionaries in Lexonomy were stored as XML data of arbitrary XML schemas, stored as plain text in an SQLite database [6] and edited using the browser-based Xonomy XML editor on the front-end [7]. The main motivation behind this decision was the emphasis on flexibility (so that users could upload dictionaries not restricted in their schemas) as well as reliance on a widely known data format (XML). Over the five years of Lexonomy development, we have however established that this (i.e. arbitrary

XML-based schemas) hinders any further Lexonomy development as a platform for automating dictionary production.

Particularly, we have established the following findings:

- **unrestricted dictionary schemas are difficult to handle by most users** For any automatic extraction of data from corpora, the user needs to manually tell what information should be included into what part of the dictionary entry, such as which entry part should contain dictionary examples, whether the samples are per-sense or per-headword etc. For users-lexicographers that are not skilled in data modelling (and we assume this is the vast majority), this is a very error-prone task, especially if they initially designed a complex dictionary schema.
- **unrestricted XML serialization is not suitable for dictionaries** While XML became standard exchange and data format in many applications, including dictionaries, it is not suitable as format that should be human-editable, it is (without any restrictions) rather difficult to process computationally in terms of data manipulation and database search and it also has been shown as not suitable for dictionary modelling [8].

While the first issue – data modelling in lexicography – is currently being addressed by the LEXIDMA consortium as a forthcoming OASIS standard [9]¹, in this paper we focus on the latter problem and describe an alternative plain text data format (NVH), its manipulation tool implemented in Python and used by Lexonomy.

2 NVH data format

NVH stands for name-value hierarchy.² It is a plain text data format significantly simpler than XML. A NVH file is a list of nodes, each node having a value and (optionally) a list of children nodes (see examples in Figures 1 and 2.)

```
node1: value of node1
  childnode1: value of childnode1
  childnode2: value of childnode2
    grandchildnode1: value of grandchildnode1
  childnode3: value of childnode3
node2: value of node2
node3: value of node3
```

Fig. 1: Structure of the NVH format in a nutshell.

¹ See <https://www.oasis-open.org/committees/lexidma/>

² See <https://namevaluehierarchy.org>

```
hw: at-c
  language: Tagalog
  lemma: at
  freq: 332418
  pos: c
  flag: ok
  sense:
    example: Kumakain siya ng prutas at gulay.
    quality: good
    example_english: She was eating fruits and vegetables.
    english: and
```

Fig. 2: A sample dictionary entry represented in NVH.

The expressiveness of the format follows from its simplicity. Each node must be placed on a separate line, it features a Python-style mandatory indentation of children nodes and each name-value pair must be separated by the colon-space character pair. No other strings bear particular semantics (value is defined as the string following the separator and ending by the newline character, it may be empty), leaving it up to the user for specification of higher-level constructs like namespaces, intra-file cross-references or external links. Obviously, NVH is much easier to parse by a program as well as much easier to read or write by a human user.

In this paper we present a processing tool for NVH that is implemented as self-contained Python script `nvh.py` and is available from the official project size of NVH.³ The script in its current version implements the following operations:

- parsing into a Python data structure
- serialization of the same structure into NVH
- search by a simple query language
- splitting by top-level node into multiple files
- merging two files in a patch-style fashion
- schema generation from an existing NVH file
- schema validation of an NVH file against a predefined schema
- export to XML
- export to JSON

The `nvh.py` script can be either imported into another Python script (which would be the typical scenario when using it for parsing and subsequent custom manipulation of the parsed content) or used from command-line where the first argument specifies the action to be taken, as we describe in detail in the following:

³ See <https://github.com/michmech/nvh/blob/master/python/python.md>

2.1 Parsing and search: `nvh.py get`

The full syntax of the command is:

```
nvh.py get file.nvh [ SELECT_FILTER [ PROJECT_FILTER ] ]
```

The selection and projection filter are optional, not using them means that the script would be parse the `file.nvh` and reprint its content. The notions of selection and projection follow the paradigm of relational algebra: a selection filter specifies which nodes should be retrieved, a projection filter specifies what parts of the selected nodes will be retrieved. Not using the projection filter implies printing the whole top-level node matching the selection criteria.

Filters use a dot-style language to identify node parts (sense translation into English given in Figure 2 would be identified as `hw.sense.english`). There may be multiple selection filters which are logically ANDed, and each node in the filter may feature one of the following operators:

- equality (`=STRING`)
- regular expression matching (`~=Python RE`)
- count (`#=`, `#>` or `#<`)

Each of the operators can be negated by prefixing it with an exclamation mark (`!`). A special selection filter taking the form of `##NUMBER` may be used only at the beginning of the list of selection filters, limiting the retrieval to the first `NUMBER` items only. An example of a search command using the data in Figure 2 would be

```
nvh.py get file.nvh 'hw.sense.example#>0.quality=good' hw.sense
```

This query would retrieve all sense (identified as `hw.sense`) from entries having at least one example marked as good quality. More examples are available in the project documentation online.⁴

2.2 Merging and patching: `nvh.py put`

The full syntax of the command is:

```
nvh.py put file.nvh patch.nvh [ REPLACE_FILTER ]
```

It merges the content of `patch.nvh` into `file.nvh` by finding shared nodes and appending any nodes and children nodes not present yet from the patch into main file. The optional dot-style replace filter may be used to select which portion of the patch (such as an entry part only) shall be merged.

2.3 Splitting: `nvh.py split`

Splitting is used to split the NVH file by top-level nodes, or to put it in dictionary terms, to generate one NVH file per dictionary entry. The command `nvh.py split file.nvh DIRECTORY` takes the `file.nvh` as input and generates individual files into `DIRECTORY`. This is useful e.g. to keep per-entry copies tracked and versioned by a file-based management system such as Git.

⁴ See <https://github.com/michmech/nvh>.

2.4 Schema generation and validation: `nvh.py genschema|checkschem`

The command `nvh.py genschema file.nvh` parses the input `file.nvh` and generates a corresponding NVH schema.⁵ An NVH schema describes valid node names and their allowed cardinality using (obligatory/optional, Kleene plus/Kleene star).

The counterpart is then represented by the `nvh.py checkschema file.nvh schema.nvh` command which validates `file.nvh` against a schema given in `schema.nvh`.

2.5 Generic exports: `nvh.py xmlexport|jsonexport`

These two commands perform generic exports to XML and JSON, respectively. The XML export transforms all nodes into XML elements bearing their value in an attribute and keeping child nodes as child XML elements. The example in Figure 2 would be transformed into an XML file as presented in Figure 3 and into JSON as presented in Figure 4.

```
<?xml version="1.0"?>
<dictionary>
  <hw v="at-v">
    <lemma v="at" />
    <language v="Tagalog" />
    <pos v="c" />
    <freq v="332418" />
    <sense>
      <example v="Kumakain siya ng prutas at gulay.">
        <quality v="good" />
        <example_english v="She was eating fruits and vegetables." />
      </example>
      <english v="and" />
    </sense>
  </hw>
</dictionary>
```

Fig. 3: Generic XML export from an NVH input of Figure 2.

3 Using NVH as a Lexonomy backbone

The flexibility of Lexonomy in terms of dictionary schemes has always been an important feature. Using NVH inside Lexonomy instead of XML enables us to

⁵ See <https://github.com/michmech/nvh/blob/master/docs/schema.md> for detailed description of the schema format.

maintain a simple text format usable for human reading and writing as well as very efficient machine processing. Data may be saved in the underlying SQLite database directly in the NVH format, individual entries but also small dictionaries can be directly searched using `nvh.py` with very low latency response (less than a second).

For large dictionaries, the JSON conversion is used for one-way encoding into JSON and using the built-in SQLite JSON indexing to store the JSON content. A simple conversion procedure has been developed to translate the query language of `nvh.py` get into SQLite SQL-JSON queries. Both NVH and JSON content is stored in the database for each entry, using JSON only for fast indexed search, but NVH for any editing. Upon every update, the JSON content is regenerated.

```
{
  "hw": [ {
    "value": "at-c",
    "children": {
      "lemma": [ {"value": "at", "children": {}} ],
      "freq": [ {"value": "332418", "children": {}} ],
      "pos": [ {"value": "c", "children": {}} ],
      "flag": [ {"value": "ok", "children": {}} ],
      "sense": [ {
        "value": "",
        "children": {
          "example": [ {
            "value": "Kumakain siya ng prutas at gulay.",
            "children": {
              "quality": [ {"value": "best", "children": {}} ],
              "example_english": [ {
                "value": "She was eating fruits and vegetables.",
                "children": {}
              } ]
            } ]
          } ]
        } ],
      "english": [ {"value": "and", "children": {}} ]
    } ]
  } ]
}
```

Fig. 4: Generic JSON export from an NVH input of Figure 2.

4 Conclusions and future development

In this paper we have presented recent development of Lexonomy consisting of replacing the XML backbone with an NVH-based one. This development is in line with the spirit of Lexonomy being a lightweight dictionary editor that can handle very large dictionaries (hundreds of thousands of complex entries) efficiently and support modern lexicographic workflow that is tightly connected to corpus data.

For Lexonomy, this particularly means to support the post-editing approach to dictionary making and Lexonomy does that by making it easy to maintain multiple versions of the dictionary, edit them simultaneously by the editorial team, split them and merge them as needed into individual editing tasks with custom editing widgets, while having a comprehensive NVH version available at every stage of the process.

In the future, more functions related to the management of the post-editing workflow are going to be added to Lexonomy, in the first place an NVH-based front-end editor is going to replace the current Xonomy-based one.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the Lindat Clarin Center. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015.

References

1. Měchura, M.B., et al.: Introducing Lexonomy: an open-source dictionary writing and publishing system. In: *Electronic Lexicography in the 21st Century: Lexicography from Scratch*. Proceedings of the eLex 2017 conference. (2017) 19–21
2. Rambousek, A., Jakubíček, M., Kosem, I.: New developments in lexonomy. *Electronic lexicography in the 21st century (eLex 2021) Post-editing lexicography (2021)* 86
3. Kilgarrieff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., Suchomel, V.: The Sketch Engine: ten years on. *Lexicography* 1 (2014)
4. Jakubíček, M., Kovář, V., Rychlý, P.: Million-Click Dictionary: Tools and Methods for Automatic Dictionary Drafting and Post-Editing. *EURALEX XIX (2021)*
5. Blahuš, M., Cukr, M., Herman, O., Jakubíček, M., Kovář, V., Medved, M.: Semi-automatic building of large-scale digital dictionaries. *Electronic lexicography in the 21st century (eLex 2021) Post-editing lexicography (2021)* 99
6. Hipp, R.D.: *SQLite* (2020)
7. Měchura, M.B.: *Building XML Editing Applications with Xonomy* (2018)
8. Měchura, M.B.: Better than XML: Towards a Lexicographic Markup Language. Available at SSRN <https://ssrn.com/abstract=4165854> (2022)
9. Tiberius, C., Krek, S., Depuydt, K., Gantar, P., Kallas, J., Kosem, I., Rundell, M.: Towards the elexis data model: defining a common vocabulary for lexicographic resources. *Electronic lexicography in the 21st century (eLex 2021) Post-editing lexicography (2021)* 91

Part II

Evaluation Methods

Are Dictionary Definitions of Verbs in Corpora?

Discovering Dead Ends in Generating Explanations of Verbs

Marie Stará

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
413827@mail.muni.cz

Abstract. Compared to nouns, there are no clear guidelines on what a definition or explanation of a verb should contain. We search corpora for the information present in Czech and English dictionaries to find out if the data is present and, thus, if it can be used as a guideline for assessing the quality of generated definitions. We show that even though a notable portion of the dictionary data is present in corpora, it does not seem frequent or specific enough to be findable with the methods we previously used.

Keywords: verb, meaning, explanation, corpora

1 Introduction

Defining a word or explaining its meaning is a task relatively easy if the word in question is a noun. Literature¹ covers the topic quite well, and gathering data to create explanations automatically is arguably a doable task (cf. [5,6]). For verbs, however, the situation is different: There is less agreement on what its definition/explanation should consist of, and thus it is more complicated to create a meaningful explanation of a verb.

We do not try to suggest the best approach to explain the meaning of a verb, merely to find out whether corpus data contain the information one can find in dictionary definitions of verbs. Subsequently, one can ask if the generated definitions can (or should) approximate the human-made definitions.

In Section 2, we list the resources we used and briefly describe the structure of our queries and the difference between Czech and English data. In Section 3, we list and discuss our results. In Section 4, we conclude that using existing dictionaries as a standard to assess the quality of generated data is not necessarily an optimal solution.

2 Method

We chose a small set of 19 verbs in Czech and English and gathered their definitions from two dictionaries, *Slovník spisovné češtiny* [1] and Macmillan

¹ For a summary (in Czech), see [6].

Table 1: The number of queries created, found, and found with low frequency for chosen verbs in Czech and English.

	queries	found	low freq.		queries	found	low freq.
bát (se)	8	4	3	fear	5	3	2
dát	19	16	1	give	7	7	3
dostat	6	4	1	get	5	5	1
existovat	3	3	0	exist	7	5	2
fungovat	5	5	5	function	7	5	1
jednat	11	8	1	act	6	5	3
ležet	4	3	1	lie	4	1	1
namlouvat	6	1	1	insinuate	2	0	-
pojistit	6	1	0	insure	3	2	1
skákat	4	2	1	jump	9	3	2
ulovit	4	3	2	hunt	10	7	3
uvést	10	4	2	initiate	4	1	1
večeřet	2	0	-	dine	1	1	1
vřít	5	4	3	boil	4	2	1
zabít	7	5	3	kill	4	2	1
začít	3	3	1	begin	4	4	2
zapomenout	6	2	2	forget	3	3	2
znemožnit	2	1	1	discredit	2	1	1
žít	4	3	3	live	4	3	0

dictionary [2], respectively. As verbs usually have a great number of senses, for verbs with multiple definitions, we used only the first three.

From these definitions, we extracted the words and phrases we considered relevant and searched for them in the czTenTen17 [3] and enTenTen13 [4] corpora². To follow the approach used in [6], the word or phrase in question had to be present in the same sentence as the given headword.

We did not try to find the exact wording of the definitions; we searched for a part of the definition at a time, often using wildcards in place of (possible) modifiers and determiners.

We used CQL queries with the following structure:

```
[lemma = "headword"] []{0,7} [lemma = "other_verb"] within < s/> |
[lemma = "other_verb"] []{0,7} [lemma = "headword"] within < s/>,
where the [lemma = "other_verb"] consisted of a single verb, a verb and its
object, or a more complicated phrase.
```

For example, (a part of) the definition for *hunt* is *to kill animals for food*. The [lemma = "other_verb"] is, in this case, replaced by: [lemma = "kill"] []{0,2} [lemma = "animal"]? [lemma = "for"] [lemma = "food"].

Although the method we chose was the same for both languages, the approaches to explaining the meaning of verbs differ in the dictionaries: In Czech, the verb is explained mostly by other (usually multiple) verbs with

² The corpora were chosen purely practically, to be big but not too much, for time is a scarce resource.

Table 2: Czech verbs, sorted by percentage of found queries and percentage of low-frequency results.

	% found	% low freq.		% found	% low freq.
existovat	100	0	existovat	100	0
fungovat	100	100	fungovat	16,7	0
začít	100	33,3	začít	84,2	6,3
dát	84,2	6,25	dát	72,7	12,5
vřít	80	75	vřít	66,7	25
ležet	75	33,3	ležet	75	33,3
ulovit	75	66,7	ulovit	100	33,3
žít	75	100	žít	50	50
jednat	72,7	12,5	jednat	40	50
zabít	71,4	60	zabít	71,4	60
dostat	66,7	25	dostat	75	66,7
bát (se)	50	75	bát (se)	50	75
skákat	50	50	skákat	80	75
znemožnit	50	100	znemožnit	100	100
uvést	40	50	uvést	16,7	100
zapomenout	33,3	100	zapomenout	33,3	100
namlouvat	16,7	100	namlouvat	50	100
pojistit	16,7	0	pojistit	75	100
večeřet	0	-	večeřet	0	-

similar meaning and, occasionally, a relevant object (noun or prepositional phrase, usually). Some of the definitions contain examples leading to listing other meanings.

In English, the definitions are more descriptive and approximate actually spoken language. These definitions usually contain an explanatory verb and its object(s) and adverbials. Some are structured as a sentence.

The different approach can be demonstrated on definitions for *hunt*. In English, the first definition is *to kill animals for food or for their skin or other parts, or for sport*; in Czech, it is *získat loven* (to obtain by hunting).

3 Results

We got at least some results for all the verbs except two (insinuate, večeřet). Generally speaking, the result for English are slightly better, meaning the percentage of queries found is higher, and the percentage of results with low frequency is lower than for Czech. nevertheless, the overall results are similar.

An overview of the results is presented in Table 1 showing the absolute number of queries created and found, together with the number of results with low frequency. As we made no thorough attempt to clear the resulting data, some of our found queries include modal or aspectual verbs, which in some cases modify other verbs not included in the definitions.

Table 3: English verbs, sorted by percentage of found queries and percentage of low-frequency results.

	% found	% low freq.		% found	% low freq.
begin	100	50	live	75	0
dine	100	100	function	71,4	20
forget	100	66,7	get	100	20
get	100	20	exist	71,4	40
give	100	42,9	give	100	42,9
act	83,3	60	hunt	70	42,9
live	75	0	begin	100	50
exist	71,4	40	boil	50	50
function	71,4	20	insure	66,7	50
hunt	70	42,9	kill	50	50
insure	66,7	50	act	83,3	60
fear	60	66,7	fear	60	66,7
boil	50	50	forget	100	66,7
discredit	50	100	jump	33,3	66,7
kill	50	50	dine	100	100
jump	33,3	66,7	discredit	50	100
initiate	25	100	initiate	25	100
lie	25	100	lie	25	100
insinuate	0	-	insinuate	0	-

Tables 2 and 3 show the results sorted according to the percentage of found queries and low-frequency results for Czech and English, respectively. We found no apparent correspondence between the number of absolute and low-frequency results.

4 Conclusion

The corpora do contain some of the dictionary definitions vocabularies. This data is, however, not frequent nor specific enough to be found by the method previously used.

When it comes to whether it makes sense to use the existing dictionary definitions as a benchmark or at least a reference point, the answer is, it depends. For Czech, we lean towards no, as the definitions are mostly lists of synonyms or synonyms with an object. (Unless, of course, we use the dictionary to check the relevance of possible synonyms.) As for English, it is something to consider.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-CLARIAH-CZ project LM2018101.

References

1. Internetová jazyková příručka, <https://prirucka.ujc.cas.cz/>, [cit. 2022-10-30]

2. Macmillan dictionary, <https://www.macmillandictionary.com/>, [cit. 2022-10-30]
3. Lexical Computing: Czech web 2017 (cstenten17), <https://www.sketchengine.eu/cstenten-czech-corpus/>, [cit. 2022-11-04]
4. Lexical Computing: English web 2013 (ententen13), <https://www.sketchengine.eu/ententen-english-corpus/>, [cit. 2022-11-04]
5. Stará, M.: Automatically created noun explanations for english. In: Aleš Horák, Pavel Rychlý, A.R. (ed.) Proceedings of the Thirteenth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2019. pp. 83–87. Tribun EU, Brno (2019)
6. Stará, M.: Automatická tvorba definic z korpusu. Diplomová práce (2019), <https://is.muni.cz/th/i9wm5/>

Evaluation of Various Approaches to Compute BLEU Metrics

Lucia Benkova  and Ľubomír Benko 

Constantine the Philosopher University in Nitra
Nitra 94901, Slovakia {lucia.benkova,lbenko}@ukf.sk

Abstract. Evaluation of machine translation (MT) performance, as the concept of quality, is closely related to the concept of optimization. Over recent decades, several approaches to evaluate MT quality have been proposed. Each approach brings new metrics for MT evaluation, and/or MT performance. The aim of our study is to show which of metrics based on precision that have been proposed so far are suitable for evaluating the quality of translation from English to Slovak in the domain of journalistic texts. We focus on the BLEU metric and its different variants that are available in the nltk libraries and the Python library. We attempt to determine which of the examined variants of the BLEU metric are redundant. The results of our research show the redundancy of BLEU-1 metric variants from the PyTorch library with respect to the newspaper style and neural MT. On the contrary, a statistically significant difference was shown by the PoS BLEU-1+1 and nltk-based BLEU-1 variants.

Keywords: Neural machine translation, Statistical machine translation, Automatic evaluation, BLEU, Slovak language, Text analysis

1 Introduction

The paper offers an evaluation of different approaches to automatic metrics for the evaluation of machine translation suitable for the Slovak language. In our previous research [1], [2], we used standard error rate and accuracy metrics such as PER, WER, TER, and BLEU. In this paper, we focus only on the state-of-art metric of accuracy, namely the BLEU-n metric from which we expect relevant results for the Slovak language and which offers open-source access to the source data and metric parameters. The BLEU metric is widely used to measure the quality of machine translation. We focus on freely Google Translate service as one of the most used online neural MT systems today.

The rest of the paper is structured as follows. The following section describes the related work of automatic metrics for MT evaluation. The third section focuses on the dataset description and methods used in the experiment. The fourth section deals with the results of the experiment where we compare various approaches to an accuracy automatic metric. The last section provides the conclusion and future work.

2 Related Work

Basic error-rate metrics include PER [3], WER [4] and TER [5] operating on the calculation of edit distance, the so-called Levenshtein distance, i.e., which provides the minimum number of edit operations (insertions, deletions or substitutions) needed to match two sequences of words. The aforementioned metrics differ from each other in their relation to word order, word position in the sentence, and translation penalty. Among the most common accuracy metrics is the BLEU-n metric [6], which, despite several flaws, is still very popular and standard within the users. BLEU-n is based on the geometric mean of the n-grams precision of length 1 to 4 and a penalty of sentence shortness (brevity penalty).

Many authors focus their research around the BLEU metrics and its variations. Benkova et al. [1] focus on the comparison of phrase-based statistical MT systems (Google SMT and mt@ec) and neural MT systems (Google NMT and eTranslation) using automatic metrics for MT evaluation from English to Slovak. The research was conducted using residuals to compare the scores of BLEU-n metrics. The results confirm the assumption of better neural MT quality regardless of the system used. Statistically significant differences between the SMT and NMT were found in favour of NMT based on all BLEU-n scores. Munkova et al. [2] focused on an evaluation of automatic measures of error rate and accuracy when validating the quality of MT output from the synthetic Slovak language to the analytical English language. They used multiple comparisons for the analysis and icon graphs to visualize the results. The results showed that all examined metrics, which are based on textual similarity, except the f-measure, are needed to be included in MT quality evaluation when analyzing MT output based on sentence. The authors [7] presented a deep evaluation and error analysis of five paraphrase generation modules of the Watson project. The results revealed the most problematic sources of errors in the generation process and helped with further improvements to the system.

Biesialska et al. [8] analysed the performance of the statistical and neural approaches to MT. They compared phrase- and neural-based MT systems and their combination. The examined language pairs were Czech–Polish and Spanish–Portuguese, and the authors used a large sample of parallel training data (they used a monolingual corpus and a pseudo-corpus). They applied back translation into their MT system and examined the scores of BLEU-n score [6]. The results showed that for the Czech–Polish language pair, the BLEU score was relatively low, which was explained by the language distance.

Almahasees [9] focused on the comparison of two MT systems, Google Translate and Microsoft Bing translator. Both systems were based on an SMT system for the English–Arabic language pair. The comparison of the MT outputs of journalistic texts was conducted using the standard automatic evaluation metric BLEU-n. The results were in favour of Google Translate, where Bing generated semantically different sentences.

3 Materials and methods

The aim of the research is to filter out the redundant metrics of automatic MT evaluation. This study can later serve as a reference to identify redundant metrics from various sets of similar metrics (BLEU, ROUGE metrics or other metrics of error rate or accuracy).

3.1 Dataset composition

We used the dataset which consists of 66 original English journalistic texts (39 354 word tokens). These texts were translated by Google Translate using SMT and NMT. Besides, texts were also translated by two professional human translators (HT) and post-edited by another professional human translator (PEMT) using our online system OSTPERE (Online System for Translation, Post-Editing, Revision, and Evaluation) [10], [11]. The translation direction was from English to Slovak, as Slovak is one of the official EU languages and contains an inflected morphology and loose word order [12]. The table 1 gives a summary of the composition of the dataset.

Table 1: Lexico-grammatical dataset composition.

Feature type	Feature name	SMT	NMT	HT	PEMT	SRC
Readability	Average sentence length	17.164	17.236	17.880	17.994	19.414
	Average word length	5.571	5.664	5.764	5.706	4.951
	Number of short sentences	487	493	466	449	413
	Number of long sentences	1557	1551	1578	1595	1631
Lexico-grammatical	Frequency of noun	9314	9365	9999	9877	8713
	Frequency of adjective	4436	4407	4659	4801	3213
	Frequency of verb	4218	4400	4437	4389	5246
	Frequency of determiner	1918	1876	1973	1971	3953
	Frequency of adposition	3735	3875	4129	4155	4680
	Frequency of proper noun	2231	2198	2165	2195	3411
	Frequency of coordinating conj.	1338	1311	1396	1334	1246
	Frequency of subordinating conj.	1352	1403	1281	1377	853
	Frequency of interjection	18	8	9	10	15
	Frequency of adverb	1307	1247	1339	1382	1653
	Frequency of pronoun	1055	1260	1417	1324	2615
	Frequency of auxiliary	1626	1299	1257	1374	2432
	Frequency of numeral	1260	1311	1195	1302	1009
	Frequency of particle	573	598	777	764	1312
	Frequency of punctuation	6668	6674	6460	6646	5370
	Frequency of other	597	561	589	511	3

3.2 Methodology

The experiment is focused on the most popular metric of accuracy- BLEU. We have taken various libraries and approaches to calculate the BLEU metrics.

The BLEU metric [6] is considered a state-of-art automatic evaluation metric. The metric is based on the geometric mean of n-gram precisions and brevity penalty (a length-based penalty). BLEU performs well at the corpus level but lags significantly at the sentence level. Lin and Och [13] applied various smoothing techniques to BLEU to obtain better results at the sentence level. Suppose we have similar n-grams for $n = 1...N$ (often $N = 4$). Let m_n be the original number of hits and m'_n be the number of hits of the modified n-gram. One smoothing technique says that if the number of matching n-grams is equal to 0, then we use a small positive value ε to replace 0 for n in the range from 1 to N .

$$m'_n = \varepsilon, \text{ if } m_n = 0.$$

There are seven smoothing techniques that are used mainly to evaluate the output based on sentences. We have focused on the second smoothing technique (the other technique's results did not yield relevant scores) that adds 1 to the number of matching n-grams and the total number of n-grams for n in the range from 2 to N .

$$l'_n = l_n + 1, \text{ for } n \text{ in } 2..N.$$

A different approach to evaluating machine translation is offered by the PoS-BLEU metric [14]. It is one of the metrics focusing on the syntactic structure of the translation output, where PoS tags are the input of the calculation instead of words.

In this experiment we will focus on the BLEU-1 metric and its variations (nltk and PyTorch library, with and without smoothing function, PoSBLEU-1+1). We expect that there will be no differences between the various BLEU-1 metrics approaches and therefore it will not play a role which approach we use in machine translation evaluation. The methodology of the experiment consists of the following steps:

1. obtaining the unstructured text data (source text) and removing the document formatting,
2. machine translation using various systems (SMT, NMT)
3. human translation of the documents,
4. post-editing of the machine translation,
5. segment alignment between the source text, machine translations, human translation and post-edited text,
6. human evaluation of examined machine translation based on model [15],
7. automatic evaluation of examined machine translation using various metrics (BLEU-1 for this experiment), where as reference text were chosen as human translation so post-edited text,
8. comparison of the translation quality based on the accuracy and translation system (SMT, NMT),
9. evaluation of obtained results.

4 Results

We have focused to identify the redundancy between various approaches to the BLEU-1 metric. We have used Python-based libraries to implement the BLEU metric. We used the library nltk, PyTorch (with and without the smoothing function) and our own function to obtain the results of POSBLEU. The POSBLEU metric needed a morphological annotation of texts, so we used the Stanza library which contains a model for the Slovak language. We have analysed the texts translated by SMT and NMT separately. Both outputs were evaluated by a human and for the SMT were identified 1574 segments that contained an error and only 470 segments were evaluated as correct. In the case of NMT, 1658 segments were correct and only 386 contained an error.

To test the global null hypotheses, we used adjusted tests for repeated measurements (Huynh-Feldt adjustment), due to the violation of the sphericity condition of the covariance matrix. If the covariance matrix sphericity condition is not satisfied, the magnitude of the type I. error increases. The epsilon represents the degree of violation of the sphericity condition. An epsilon equal to one represents the satisfaction of the condition. Conversely, the smaller it is, the more the sphericity condition is violated.

When testing the global null hypotheses, epsilon values were less than one (Table 2). In the case of SMT, null hypotheses are rejected with 99.9% confidence (at the 0.001 significance level). The hypotheses assert that group segment accuracy does not depend on variations in BLEU-1 accuracy metrics and combinations of BLEU-1 and segment accuracy factors (manual evaluation 0/1).

Similarly, in the case of the NMT, it has been shown that the accuracy of the segments studied depends on the variation of the BLEU-1 accuracy metrics. In contrast, the dependence on the combination of BLEU-1 and segment accuracy factors (manual evaluation 0/1) was not confirmed.

In terms of multiple comparisons (Table 3), we have identified three homogeneous groups (**** - $p > 0.05$) in the degree of accuracy of the examined segments. A statistically significant difference in segment accuracy rates was demonstrated between POSBLEU_1+1 and the others, and similarly between

Table 2: Huynh-Feldt adjustment for BLEU-1 and segment accuracy for (a) SMT and (b) NMT.

(a)				
NMT=0	H-F Epsilon	H-F Adj. df1	H-F Adj. df2	H-F Adj. p
BLEU-1	0.5087	1.5260	3116.1310	0.0000
BLEU-1*Evaluation_Error	0.5087	1.5260	3116.1310	0.000
(b)				
NMT=1	H-F Epsilon	H-F Adj. df1	H-F Adj. df2	H-F Adj. p
BLEU-1	0.5558	1.6675	3404.9990	0.0000
BLEU-1*Evaluation_Error	0.5558	1.6675	3404.9990	0.8424

Table 3: Multiple comparisons for various BLEU-1 metrics and segment accuracy for (a) SMT and (b) NMT.

(a) NMT=0				
BLEU-1	Mean	1	2	3
PyTorch_BLEU-1_smooth	0.504	****		
PyTorch_BLEU-2	0.504	****		
BLEU-1	0.626		****	
POSBLEU-1+1	0.719			****
(b) NMT=1				
BLEU-1	Mean	1	2	3
PyTorch_BLEU-1_smooth	0.519	****		
PyTorch_BLEU-2	0.519	****		
BLEU-1	0.664		****	
POSBLEU-1+1	0.743			****

BLEU-1 and the other metrics ($p < 0.05$). On the other hand, a statistically significant difference was not identified between the PyTorch metrics. The results are the same for both translation systems, the expected higher accuracy rates were achieved for NMT. From this point of view, the redundant metric will be precisely one of these PyTorch metrics.

The results showed us that the PyTorch metrics are redundant. In this case, the smoothing function that was introduced to improve the evaluation based on segments did not produce different results than the corpus-based BLEU-1 metric from the PyTorch library. In the future, we can omit the smoothing function variant of the BLEU-1 metric.

5 Conclusion

The paper deals with the metrics of the automatic MT evaluation and is a basis for our future experiments. We have introduced a methodology to filter out the redundant metrics that were experimented on using the BLEU-1 metric. This will be expanded in future work that will deal with a greater number of automatic metrics, that will be grouped based on related characteristics. We would also like to compare newer metrics, like ChrF++ [16], BEER [17], LEPOR [18], COMET [19], with older like NIST [20], ROUGE [21], METEOR [22]. The aim is to select the most appropriate automatic metrics for evaluating MT output into Slovak. In this paper, we have shown that various approaches to calculate the BLEU-1 metric show significant differences. However, the use of the smoothing function does not produce significantly different results than using the corpus-based BLEU-1 metric.

Acknowledgement This work was supported by the Slovak Research and Development Agency under the contract No. APVV-18-0473 and by the projects UGA VII/2/2022 and UGA VII/1/2022.

References

1. Benkova, L., Munkova, D., Benko, L., Munk, M.: Evaluation of English–Slovak Neural and Statistical Machine Translation. *Applied Sciences*. 11, (2021). <https://doi.org/10.3390/app11072948>.
2. Munkova, D., Hajek, P., Munk, M., Skalka, J.: Evaluation of Machine Translation Quality through the Metrics of Error Rate and Accuracy. *Procedia Comput Sci*. 171, 1327–1336 (2020). <https://doi.org/10.1016/j.procs.2020.04.142>.
3. Tillmann, C., Vogel, S., Ney, H., Zubiaga, A., Sawaf, H.: Accelerated DP based search for statistical translation. In: *European Conference on Speech Communication and Technology*. pp. 2667–2670. Rhodes, Greece (1997).
4. Snover, M., Dorr, B., Schwartz, R., Micciulla, L., Makhoul, J.: A study of translation edit rate with targeted human annotation. In: *Proceedings of Association for Machine Translation in the Americas*. pp. 223–231 (2006).
5. Nießen, S., Och, F.J., Leusch, G., Ney, H.: An evaluation tool for machine translation: Fast evaluation for MT research. In: *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC-2000)*. pp. 39–45 (2000).
6. Papineni, K., Roukos, S., Ward, T., Zhu, W.: BLEU: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. pp. 311–318. , Philadelphia (2002).
7. Burgerová, V., Horák, A.: Evaluation and Error Analysis of Rule-based Paraphrase Generation for Czech. In: Horák, A., Rychlý, P., and Rambousek, A. (eds.) *Proceedings of the Thirteenth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2019*. pp. 33–39. Tribun EU, Brno (2019).
8. Biesialska, M., Guardia, L., Costa-jussa, M.R.: The TALP-UPC System for the WMT Similar Language Task: Statistical vs Neural Machine Translation. In: *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*. pp. 185–191. Association for Computational Linguistics, Florence, Italy (2019). <https://doi.org/10.18653/v1/W19-5424>.
9. Almahasees, Z.M.: Assessing the Translation of Google and Microsoft Bing in Translating Political Texts from Arabic into English. *International Journal of Languages, Literature and Linguistics*. 3, 1–4 (2017). <https://doi.org/10.18178/ijll.2017.3.1.100>.
10. Munková, D., Munk, M., Benko, L., Absolon, J.: From Old Fashioned “One Size Fits All” to Tailor Made Online Training. In: *Advances in Intelligent Systems and Computing*. pp. 365–376. Springer Verlag (2020). https://doi.org/10.1007/978-3-030-11932-4_35.
11. Munková, D., Kapusta, J., Drlík, M.: System for Post-Editing and Automatic Error Classification of Machine Translation. In: *DIVAI 2016: 11th International Scientific Conference on Distance Learning in Applied Informatics, Sturovo, May 2 – 4, 2016*. pp. 571–579. Wolters Kluwer, ISSN 2464-7489, Sturovo (2016).
12. Kosta, P.: Targets, Theory and Methods of Slavic Generative Syntax: Minimalism, Negation and Clitics. In: Kempgen, Sebastian / Kosta, Peter / Berger, Tilman / Gutschmidt, Karl (eds.). *Slavic Languages. Slavische Sprachen. An International Handbook of their Structure*. In: Kempgen, S., Kosta, P., Berger, T., and Gutschmidt, K. (eds.) *Slavic Languages. Slavische Sprachen. An International Handbook of their Structure, their History and their Investigation. Ein internationales Handbuch ihrer Struktur, ihrer Geschichte und ihrer Erforschung*. pp. 282–316. Berlin, New York: Mouton. de Gruyter (2009).
13. Lin, C.-Y., Och, F.J.: Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In: *Proceedings of the*

- 42nd Annual Meeting on Association for Computational Linguistics - ACL '04. pp. 605-es. Association for Computational Linguistics, Morristown, NJ, USA (2004). <https://doi.org/10.3115/1218955.1219032>.
14. Popović, M., Ney, H.: Syntax-oriented evaluation measures for machine translation output. In: Proceedings of the Fourth Workshop on Statistical Machine Translation - StatMT '09. p. 29. Association for Computational Linguistics, Morristown, NJ, USA (2009). <https://doi.org/10.3115/1626431.1626435>.
 15. Vaňko, J.: Kategoriálny rámec pre analýzu chýb strojového prekladu. In: Munkova, D. and Vaňko, J. (eds.) *Mýliť sa je ľudské (ale aj strojové)*. pp. 83–100. UKF v Nitre, Nitra (2017).
 16. Popović, M.: chrF: character n-gram F-score for automatic MT evaluation. In: Proceedings of the Tenth Workshop on Statistical Machine Translation. pp. 392–395. Association for Computational Linguistics, Stroudsburg, PA, USA (2015). <https://doi.org/10.18653/v1/W15-3049>.
 17. Stanojević, M., Sima'an, K.: Evaluating MT systems with BEER. The Prague Bulletin of Mathematical Linguistics. 104, 17–26 (2015). <https://doi.org/10.1515/pralin-2015-0010>.
 18. Han, A.L.F., Wong, D.F., Chao, L.S.: LEPOR: A Robust Evaluation Metric for Machine Translation with Augmented Factors. In: Proceedings of COLING 2012: Posters. pp. 441–450. The COLING 2012 Organizing Committee, Mumbai, India (2012).
 19. Rei, R., Stewart, C., Farinha, A.C., Lavie, A.: COMET: A Neural Framework for MT Evaluation. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 2685–2702. Association for Computational Linguistics, Stroudsburg, PA, USA (2020). <https://doi.org/10.18653/v1/2020.emnlp-main.213>.
 20. Doddington, G.: Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. 138–145 (2002).
 21. Lin, C.-Y.: ROUGE: A Package for Automatic Evaluation of Summaries. In: Text Summarization Branches Out. pp. 74–81. Association for Computational Linguistics, Barcelona, Spain (2004).
 22. Lavie, A., Denkowski, M.: The Meteor metric for automatic evaluation of machine translation. Machine Translation. 23, 105–115 (2009).

Compressed FastText Models for Czech Tagger

Zuzana Nevěřilová

Natural Language Processing Centre
Faculty of Informatics
Botanická 68a, Brno, Czech Republic

Abstract. We are building a new tagger for the Czech language that uses two models: the FastText model for word embeddings and a neural network that assigns tags to tokens. In the deployment, we are struggling with model sizes. Since the model size is a common obstacle in various tasks, several compression methods exist. Authors of the methods often claim that the impact on model performance is minimal. However, the evaluation is done on the two tasks the word embeddings are evaluated on: word analogy and word similarity. No information is provided for the evaluation of subsequent tasks.

In this paper, we have trained a FastText word embedding model on more recent data. We retrained the tagger with the same parameters using compressed and uncompressed variants of the original FastText model and the new one. After comparing the results, we can see quantization methods are suitable, possibly together with pruning, without significant impact on the tagger performance. The precision dropped by 0.1 percentage point only in quantized models. All tested compression methods reduce the model size 10–100 times.

Keywords: model compression, FastText, embedding evaluation, Czech tagger

1 Introduction

Many applications use word embeddings of different flavors. In some applications, developers struggle with hardware requirements. So model size reduction or compression is a relevant topic. In [11], we proposed a neural tagger for Czech that uses FastText embeddings. The main advantage of FastText is the use of subwords. It solves the out-of-vocabulary problem (OOV) that is significant for highly inflectional languages. We trained a tagging model, and the two models must be loaded together in memory, consuming more than 7GB of memory. From the two models, the FastText model is much larger; therefore, it seems reasonable to reduce it, preferably with no impact on the performance of the tagger.

In Section 2, we describe in short the neural tagger for Czech. Section 3 describes the original embeddings and training of the new embeddings. In Section 4, we describe in short different compression methods and the model sizes without and with different compression methods. Section 5 describes the

evaluation scheme, and Section 6 summarizes the evaluation of the tagger with different models.

2 Neural Tagger for Czech

We proposed a neural network with the following architecture: The input text is tokenized, and all tokens are converted to vectors using the FastText library. The input layer consists of a padded sequence of FastText embeddings. Two Bi-LSTM layers with spatial dropouts follow, and the output layer is time-distributed.

We converted the tagging task into a multiclass classification. We split each tag into *attributes* such as part-of-speech or grammatical gender. The classifier has to predict a set of one or more attributes for each token. We trained the tagger on 300k sentences from the csTenTen17 corpus [12]. The corpus is created from the web, it contains both standard and informal Czech. The sentences were tagged with the *desamb* tagger and used the same tagset as *desamb* and the morphological analyzer *majka* [13]. In [6], the authors explain all possible tags.

Since the number of classes affects the neural model size, we discarded the attribute groups or attributes that can be found in an external dictionary or rarely occur in the data:

- verb aspect (the *a* attribute),
- adverb type (the *t* attribute) such as time, respect, reason
- pronoun subclassification (the *x* attribute) such as personal or possessive
- pronoun type (the *y* attribute) such as interrogative or relative
- negation (the *e* attribute)
- stylistic subclassification (the *w* attribute)
- the *gR* attribute (family gender)

We merged all punctuation marks under one tag. After this preprocessing of the input data, we reduced the number of possible attributes to 44 (the neural network output size). Used attribute groups are:

- *k* – part of speech
- *g* – gender (masculine, feminine, neutral, masculine inanimate)
- *n* – number (plural, singular)
- *c* – case (nominative, genitive, etc., note that Czech has seven cases)
- *m* – verb tense (present, infinitive, past participle, etc.)
- *p* – person (relevant for pronouns and verbs)
- *d* – the degree of adjectives and adverbs (positive, comparative, superlative)
- *x* – punctuation

The neural network performs multiclass attribute classification. The limitation is that some classes are exclusive, notably, only one attribute of a group can be assigned to a token. We select the attribute with the highest probabilities among mutually exclusive candidates.

We also use a threshold (currently 0.5) for the probabilities. The consequence is that some tokens have assigned only a subset of grammatical tags.

The intended use of the tagger is in the pipeline, as depicted in Figure 1. The tagger does not provide lemmata, so we have to use a morphological dictionary and a guesser of OOV words. The morphological dictionary can help in case of incomplete tags. Both tasks – filling in missing tags and providing the lemmata – are planned for the postprocessing step.

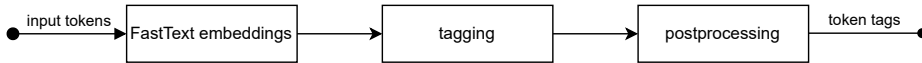


Fig. 1: Tagger processing pipeline

3 Different Embedding Models for the Tagger

In the following experiments, we preserve the tagger architecture and all parameters. The only thing that changes is the embedding model. First, we experimented with the pre-trained model for Czech¹. Second, we experimented with different compression methods. Third, we trained our FastText model for the Czech language from two different language resources: Wikipedia and the SYN corpus. The advantage of the resulting model is that it uses more recent data. On the other hand, the model size is bigger than the original pre-trained model size, even though the training data were smaller. We experimented with the compression of this new model as well. With each FastText model, we retrained the neural tagger for Czech and evaluated it on 10k sentences not present in the training data. The goal is not to have the best parameters for the tagger but to see the impact of different FastText models on tagger performance.

3.1 Pre-trained FastText model for Czech

In [5], the authors present pre-trained FastText models for 157 languages. The model for Czech was created from two sources: Wikipedia² and Common-Crawl³. The former is a collection of high-quality texts, however, the corpus size is relatively small: Czech Wikipedia corpus contained 179 million tokens in 2017, and 785k appeared at least five times. The latter contained 13 billion tokens in 2017, and the vocabulary size of the model was 8.7 million.

3.2 New FastText model for Czech

We decided to train the FastText embeddings from scratch. We used the Wikipedia corpus, which has grown to 218 million tokens, from which 863k ap-

¹ Available at <https://fasttext.cc/docs/en/crawl-vectors.html>

² <https://www.wikipedia.org/>

³ <https://commoncrawl.org/>

peared at least five times. For the processing of the Wikipedia data, we used the Wikicorpora package⁴.

Instead of CommonCrawl, we decided to use the SYN corpus, version 9. The SYN corpus [9] contains all synchronic written corpora of the SYN series. The names of the SYN corpora contain the year the corpus was made (the complete SYN consists of collections from SYN2000 to SYN2020). It does not mean the date the texts were created, the oldest texts are books from the 1900s. A large part of SYN is journalistic texts. Overall, the SYN corpora contain mainly formal, grammatically correct texts.

The SYN corpus, v.9, contains 5.9 billion tokens, with 10.8 million tokens vocabulary. Note that the vocabulary size of the model is smaller since tokens with small frequencies are discarded. The vocabulary size of the SYN model is 3 million tokens. The SYN corpus is therefore about half of the size of the 2017 CommonCrawl used in the original work.

For training, we used the same parameters as were described by authors of the original FastText model published in [5] and the FastText documentation⁵: minimum and maximum length of character n-grams set to 5 (the `minn` and `maxn` options), vector dimension set to 300, 10 epochs, negatives sampled 5 (the `neg` option).

4 Embedding Compression Methods

Because of the word embedding size and intended use in small devices, compression methods have been an emerging issue since 2014.

These include:

- reduction of vocabulary size
 - discarding infrequent tokens
 - discarding non-discriminative tokens after training
- feature selection
 - discarding features that do not influence the classifier much
- vector dimensionality reduction
 - discarding a fixed proportion of the vector dimension
- matrix decomposition
- quantization – an approximation of vectors by quantized values
- hashing
- reduction of vocabulary size

The methods that discard some information are similar to hyperparameter tuning because they have to be fine-tuned for a particular dataset and task. Hence, they are not easily transferable to other tasks. Some of the compression methods are described in [1]. More sophisticated methods often aim to be

⁴ <https://github.com/effa/wikicorpora>

⁵ <https://fasttext.cc/docs/en/options.html>

more universal. For example, matrix decomposition is used in Distilled embeddings [10] proven to outperform the state-of-the-art results in machine translation. Quantization is a method that approximates real values to centroid values. Product quantization, as described in [8], can encode the same information about nearest neighbors with significantly lower memory usage. Floret embeddings [2] recently added to SpaCy use the MurmurHash algorithm that encodes vectors in several hash tables with a smaller number of hashes. As a result, the Floret embeddings require less memory, resulting in vector similarity comparable to FastText.

Not all of the above methods can be applied to FastText embeddings. The difference is that FastText encodes vectors also for subwords. FastText returns a vector from the vector dictionary if it is present or a sum of vectors of all n-grams of the word. Using this method, FastText can deal quite well with misspellings and rare forms in inflectional languages.

The original FastText library does not support compression for unsupervised models, even though it supports compression for other models [7]. We used the `compress_fasttext`⁶ Python module for our experiments. We also wanted to check whether feature selection and quantization – the methods recommended in [4] by the author of the `compress_fasttext` library – are the most suitable methods for our task.

4.1 Parameters of Uncompressed and Compressed Models

The FastText model compressed using pruning reduce vocabulary size to 20k and n-gram size to 100k tokens. The quantization parameters are the default, 255 centroids, and the quantization dimension equal to 100. Table 1 shows the model sizes.

Table 1: Model descriptions and sizes

model name	description	model size
<code>cc.cs.300.bin</code>	Original uncompressed FastText model	6.8GB
<code>cc.cs.300_prune_freq</code>	Feature selection without quantization	70MB
<code>cc.cs.300_prune_freq_pq</code>	Feature selection with quantization	14MB
<code>cc.cs.300_quantize</code>	Quantization	426MB
<code>cc.cs.300_svd</code>	Matrix decomposition (SVD)	273MB
<code>syn_wiki.bin</code>	Uncompressed new FastText model	9.9GB
<code>syn_wiki_prune_freq</code>	Feature selection without quantization	70MB
<code>syn_wiki_prune_freq_pq</code>	Feature selection with quantization	14MB
<code>syn_wiki_quantize</code>	Quantization	587MB
<code>syn_wiki_svd</code>	Matrix decomposition (SVD)	381MB

⁶ <https://github.com/avidale/compress-fasttext>

5 Evaluation methods

The FastText models were evaluated on the word analogy task. This work does not evaluate the trained models on the analogies. Instead, we evaluate the subsequent tagging. While the word analogy task shares some aspects with the tagging (e.g., the analogous words are the same part of speech), some aspects may differ (e.g., the grammatical gender is not always relevant for the analogies but for the tagging).

We retrained the neural tagger with all FastText models, every time with the same hyperparameters:

- batch size: 128
- epochs: 15
- initial learning rate: 0.002
- decay: 0.00013
- max. sentence length: 20

For each model, we compared the predicted attributes to the ground truth. We classified the errors into categories defined in the MUC evaluation scheme [3], with no possible partial match. We calculated the number of attributes in each of the categories:

- COR: correct
- INC: incorrect, the attribute group was predicted, but it was different
- MIS: missing, the attribute was not predicted
- SPU: spurious, the attribute was predicted, but there was no attribute in the ground truth

In addition, we counted all cases where the tag was completely and correctly predicted (exact match). The validation data are 10k Czech sentences from the csTenTen17 corpus. The sentences contain 78,815 tokens.

6 Results

In this section, we show detailed results for all models listed in Table 1. The attribute meanings are listed in Section 2. Precision and recall are calculated according to the MUC evaluation scheme as follows:

$$\begin{aligned} ACT &= COR + INC + SPU & P &= \frac{COR}{ACT} = \frac{TP}{TP+FP} \\ POS &= COR + INC + MIS & R &= \frac{COR}{POS} = \frac{TP}{TP+FN} \end{aligned}$$

In Table 2, we present the overall precision and recall for all models, together with the percentage of exact matches (complete and correct tags). More detailed results are available in the Appendix.

The results show that the new model outperforms the original FastText model. This is surprising since we supposed the training data from Common Crawl would be closer to the evaluation data from csTenTen17 than the SYN corpus. Moreover, the training data were smaller in the case of the new model.

Table 2: Summary of precision and recall for compressed and uncompressed models

model name	size	precision	recall	% exact matches
cc.cs.300.bin	6.8GB	0.93	0.91	79.52
cc.cs.300_prune_freq	70MB	0.93	0.89	77.80
cc.cs.300_prune_freq_pq	14MB	0.93	0.89	77.12
cc.cs.300_quantize	426MB	0.93	0.90	78.88
cc.cs.300_svd	273MB	0.92	0.86	75.10
syn_wiki.bin	9.9GB	0.95	0.93	83.40
syn_wiki_prune_freq	70MB	0.94	0.92	82.33
syn_wiki_prune_freq_pq	14MB	0.94	0.92	82.07
syn_wiki_quantize	587MB	0.95	0.93	83.11
syn_wiki_svd	381MB	0.93	0.91	79.84

A downside is the new model is much larger. The evaluation of compressed models indicates that compression methods do not decrease performance much. It can be seen that matrix decomposition is the less appropriate method. Another observation is that quantization does not affect model performance, but it significantly affects the model size. Therefore, we recommend using quantization.

7 Conclusion and Future Work

Quantized models without pruning have the best results among the compressed models. The size of the quantized model is one order of magnitude higher than that of the pruned models, however, still one order of magnitude lower than the uncompressed models.

For the pipeline, we can safely use the quantized models, either with or without pruning. Further work includes postprocessing with the morphological analyzer that can fill the word lemmata and missing attributes in many cases. The last component of the tagger pipeline that has to be developed is a guesser for OOV tokens.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-CLARIAH-CZ project LM2018101.

References

1. Andrews, M.: Compressing word embeddings. CoRR **abs/1511.06397** (2015), <http://arxiv.org/abs/1511.06397>
2. Boyd, A., Warmerdam, V.D.: floret: lightweight, robust word vectors (2022), <https://explosion.ai/blog/floret-vectors>, [Online; posted AUG 23, 2022]

3. Chinchor, N., Sundheim, B.: MUC-5 evaluation metrics. In: Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993 (1993), <https://aclanthology.org/M93-1007>
4. Dale, D.: Compressing unsupervised fasttext models (December 2021), <https://towardsdatascience.com/eb212e9919ca>, [Online; posted 12-December-2021]
5. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki, Japan (May 2018), <https://aclanthology.org/L18-1550>
6. Jakubíček, M., Kovář, V., Šmerk, P.: Czech morphological tagset revisited. In: Horák, R. (ed.) Proceedings of Recent Advances in Slavonic Natural Language Processing 2011. pp. 29–42. Tribun EU, Brno (2011), <https://nlp.fi.muni.cz/raslan/2011/paper05.pdf>
7. Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T.: Fasttext.zip: Compressing text classification models (2016), <http://arxiv.org/abs/1612.03651>, cite arxiv:1612.03651Comment: Submitted to ICLR 2017
8. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(1), 117–128 (2011). <https://doi.org/10.1109/TPAMI.2010.57>
9. Křen, M., Cvrček, V., Henyš, J., Hnátková, M., Jelínek, T., Koček, J., Kovářková, D., Křivan, J., Milička, J., Petkevič, V., Procházka, P., Skoumalová, H., Šindlerová, J., Škrabal, M.: SYN v9: large corpus of written czech (2021), <http://hdl.handle.net/11234/1-4635>, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University
10. Lioutas, V., Rashid, A., Kumar, K., Haidar, M.A., Rezagholizadeh, M.: Distilled embedding: non-linear embedding factorization using knowledge distillation. CoRR **abs/1910.06720** (2019), <http://arxiv.org/abs/1910.06720>
11. Nevěřilová, Z., Stará, M.: Neural tagger for czech language: Capturing linguistic phenomena in web corpora. In: Horák, A., Rychlý, P., Rambousek, A. (eds.) The 13th Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2019, Karlova Studanka, Czech Republic, December 6-8, 2019. pp. 23–32. Tribun EU (2019), <http://nlp.fi.muni.cz/raslan/2019/paper10-neverilova.pdf>
12. Suchomel, V.: cstent17, a recent czech web corpus. In: Aleš Horák, P.R., Rambousek, A. (eds.) Proceedings of the Twelfth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2018. pp. 111–123. Tribun EU, Brno (2018), <https://nlp.fi.muni.cz/raslan/2018/paper10-Suchomel.pdf>
13. Šmerk, P., Rychlý, P.: Majka – rychlý morfologický analyzátor. Tech. rep., Masarykova univerzita (2009), <http://nlp.fi.muni.cz/ma/>

Appendix

Table 3: Original (Wiki+CommonCrawl) FastText Model: Uncompressed Model

Attr	COR	INC	MIS	SPU	ACT	POS	P	R
k	74745	2620	1450	0	77365	78815	0.97	0.95
g	28026	3242	3349	408	31676	34617	0.88	0.81
n	39825	2068	584	610	42503	42477	0.94	0.94
c	34346	4096	2219	524	38966	40661	0.88	0.84
m	11756	249	561	173	12178	12566	0.97	0.94
d	11090	122	760	889	12101	11972	0.92	0.93
p	7314	202	929	66	7582	8445	0.96	0.87
x	11420	0	2	167	11587	11422	0.99	1.0
Total	218522	12599	9854	2837	233958	240975	0.93	0.91

Table 4: Original (Wiki+CommonCrawl) FastText Model: Compression using feature selection with product quantization (recommended method)

Attr	COR	INC	MIS	SPU	ACT	POS	P	R
k	73947	3163	1705	0	77110	78815	0.96	0.94
g	26745	3328	4544	506	30579	34617	0.87	0.77
n	39082	2471	924	736	42289	42477	0.92	0.92
c	33675	4355	2631	625	38655	40661	0.87	0.83
m	11461	263	842	180	11904	12566	0.96	0.91
d	10816	151	1005	912	11879	11972	0.91	0.9
p	7020	267	1158	75	7362	8445	0.95	0.83
x	11420	0	2	81	11501	11422	0.99	1.0
Total	214166	13998	12811	3115	231279	240975	0.93	0.89

Table 5: Original (Wiki+CommonCrawl) FastText Model: Compression using feature selection without product quantization

Attr	COR	INC	MIS	SPU	ACT	POS	P	R
k	74044	2821	1950	0	76865	78815	0.96	0.94
g	27035	3220	4362	524	30779	34617	0.88	0.78
n	39140	2412	925	735	42287	42477	0.93	0.92
c	33352	4221	3088	529	38102	40661	0.88	0.82
m	11546	281	739	207	12034	12566	0.96	0.92
d	10768	143	1061	815	11726	11972	0.92	0.9
p	6958	272	1215	69	7299	8445	0.95	0.82
x	11420	0	2	167	11587	11422	0.99	1.0
Total	214263	13370	13342	3046	230679	240975	0.93	0.89

Table 6: Original (Wiki+CommonCrawl) FastText Model: Quantization

Attr	COR	INC	MIS	SPU	ACT	POS	P	R
k	74309	2605	1901	0	76914	78815	0.97	0.94
g	27649	3416	3552	475	31540	34617	0.88	0.8
n	39726	2097	654	627	42450	42477	0.94	0.94
c	34227	4115	2319	516	38858	40661	0.88	0.84
m	11724	284	558	165	12173	12566	0.96	0.93
d	10768	133	1071	763	11664	11972	0.92	0.9
p	7136	253	1056	68	7457	8445	0.96	0.84
x	11420	0	2	167	11587	11422	0.99	1.0
Total	216959	12903	11113	2781	232643	240975	0.93	0.9

Table 7: Original (Wiki+CommonCrawl) FastText Model: Matrix decomposition (SVD)

Attr	COR	INC	MIS	SPU	ACT	POS	P	R
k	72963	3367	2485	0	76330	78815	0.96	0.93
g	25379	4097	5141	491	29967	34617	0.85	0.73
n	38176	2403	1898	778	41357	42477	0.92	0.9
c	32743	4411	3507	439	37593	40661	0.87	0.81
m	10665	606	1295	253	11524	12566	0.93	0.85
d	10034	344	1594	1243	11621	11972	0.86	0.84
p	6456	290	1699	204	6950	8445	0.93	0.76
x	11418	0	4	85	11503	11422	0.99	1.0
Total	207834	15518	17623	3493	226845	240975	0.92	0.86

Table 8: Wiki+SYN FastText Model: Uncompressed Model

Attr	COR	INC	MIS	SPU	ACT	POS	P	R
k	75650	2109	1056	0	77759	78815	0.97	0.96
g	30449	2576	1592	336	33361	34617	0.91	0.88
n	40427	1753	297	363	42543	42477	0.95	0.95
c	35296	3995	1370	533	39824	40661	0.89	0.87
m	12085	41	440	156	12282	12566	0.98	0.96
d	11492	34	446	769	12295	11972	0.93	0.96
p	8064	28	353	48	8140	8445	0.99	0.95
x	11420	0	2	21	11441	11422	1.00	1.00
Total	224883	10536	5556	2226	237645	240975	0.95	0.93

Table 9: Wiki+SYN FastText Model: Compression using feature selection with product quantization

Attr	COR	INC	MIS	SPU	ACT	POS	P	R
k	75091	2318	1406	0	77409	78815	0.97	0.95
g	29757	2680	2180	357	32794	34617	0.91	0.86
n	39825	1955	697	456	42236	42477	0.94	0.94
c	34858	4107	1696	588	39553	40661	0.88	0.86
m	11888	75	603	158	12121	12566	0.98	0.95
d	11364	33	575	728	12125	11972	0.94	0.95
p	7871	29	545	41	7941	8445	0.99	0.93
x	11413	0	9	62	11475	11422	0.99	1.0
Total	222067	11197	7711	2390	235654	240975	0.94	0.92

Table 10: Wiki+SYN FastText Model: Compression using feature selection without product quantization

Attr	COR	INC	MIS	SPU	ACT	POS	P	R
k	75130	2491	1194	0	77621	78815	0.97	0.95
g	29666	2713	2238	321	32700	34617	0.91	0.86
n	39939	1972	566	468	42379	42477	0.94	0.94
c	34791	4030	1840	661	39482	40661	0.88	0.86
m	11904	73	589	142	12119	12566	0.98	0.95
d	11341	32	599	693	12066	11972	0.94	0.95
p	7942	44	459	67	8053	8445	0.99	0.94
x	11413	0	9	43	11456	11422	1.0	1.0
Total	222126	11355	7494	2395	235876	240975	0.94	0.92

Table 11: Wiki+SYN FastText Model: Quantization

Attr	COR	INC	MIS	SPU	ACT	POS	P	R
k	75649	2269	897	0	77918	78815	0.97	0.96
g	30289	2646	1682	308	33243	34617	0.91	0.87
n	40310	1760	407	493	42563	42477	0.95	0.95
c	34966	3931	1764	434	39331	40661	0.89	0.86
m	12099	50	417	160	12309	12566	0.98	0.96
d	11445	37	490	681	12163	11972	0.94	0.96
p	8037	34	374	47	8118	8445	0.99	0.95
x	11420	0	2	86	11506	11422	0.99	1.0
Total	224215	10727	6033	2209	237151	240975	0.95	0.93

Table 12: Wiki+SYN FastText Model: Matrix decomposition (SVD)

Attr	COR	INC	MIS	SPU	ACT	POS	P	R
k	74744	2585	1486	0	77329	78815	0.97	0.95
g	27629	3634	3354	350	31613	34617	0.87	0.8
n	39902	2117	458	949	42968	42477	0.93	0.94
c	34411	4260	1990	697	39368	40661	0.87	0.85
m	11849	241	476	192	12282	12566	0.96	0.94
d	10979	169	824	822	11970	11972	0.92	0.92
p	7607	187	651	84	7878	8445	0.97	0.9
x	11419	0	3	135	11554	11422	0.99	1.0
Total	218540	13193	9242	3229	234962	240975	0.93	0.91

Blooming Onion: Efficient Deduplication through Approximate Membership Testing

Ondřej Herman^{1,2}

¹ Faculty of Informatics, Masaryk University
Botanická 68a, 612 00 Brno, Czech Republic
`xherman1@fi.muni.cz`

² Lexical Computing s.r.o.
Botanická 68a, 612 00 Brno, Czech Republic
`ondrej.herman@sketchengine.eu`

Abstract. Deduplication of source text is an important step in corpus building. Maximum corpus sizes have been grown significantly, along with the requirements for computing resources required for processing them. This article explores reducing the cost of deduplication by applying approximate membership testing using Bloom filtering.

Keywords: deduplication, text corpora, Bloom filter

1 Introduction

Deduplication is an essential step in the preparation of text corpora for many downstream tasks in natural language processing. For example, in the process of training unsupervised machine learning models, repeated instances of the same data can cause significant biases. In fulltext search applications, the end users do not want to see repeated results for a single query, but a balanced representation of the source corpus. In linguistic applications, repeated text in the source data influences the statistics derived from it and reduces the quality and representativeness of the result.

Duplicates appear for many reasons in text data. Humans like to copy. This happens on many levels. Citations, boilerplate, or outright spam are common reasons. Data obtained from the Web is rife with repeated text in the main content, but also advertisements, context management system artifacts and links to the same, repeated content. The repeated content is not always an exact copy, but is sometimes changed slightly, to escape detection, or simply due to errors, so detection of exact instances is not enough for practical use, near duplicates also need to be considered.

1.1 Onion

The tool we use for text deduplication for the building of corpora at Sketch Engine ([6]) is Onion ([7]). Onion (ONe Instance ONLY) works on the vertical text

format. At this stage, the text has already been tokenized and is represented as a single line per token, possibly with additional data for the same token separated by TAB characters on the same line, such as lemmata or part-of-speech tags. The text is segmented at least into documents and paragraphs, delimited by `<doc>`, `</doc>` and `<p>`, `</p>` markers. Other markers, such as `<s>` for describing sentence boundaries, may be present. For example, the beginning of the Susanne corpus in the vertical text format:

```
<doc file="A01" n="1">
<p>
<s>
The      the      AT
Fulton   Fulton   NP1s
County   county   NNL1cb
Grand    grand    JJ
Jury     jury     NN1c
said     say      VVDv
Friday   Friday   NPD1
an       an       AT1
investigation  investigation  NN1n
```

Onion works by detecting and discarding duplicate or near-duplicate paragraphs. Paragraph is split into overlapping sequence of tuples of words, called shingles. For example, shingles of length 3 in the above text would be *(The, Fulton, County)*, *(Fulton, County, Grand)*, *(County, Grand, Jury)* and so on. A paragraph is considered to be a duplicate if the proportion of already seen shingles contained within it is larger than a specific threshold. For our purposes, we set this threshold to 50 % and the shingle length to 7.

Onion works by storing the hashes of all already seen shingles in a hash table, and therefore the memory requirements can be quite significant for large corpora. In the following, I explore the possibility of replacing the hash table, which stores the exact hashes for every shingle, by an approximate data structure. This can have a significant effect on memory requirements, but only a small and predictable effect on the precision of the threshold check.

2 Approximate membership testing

Membership testing is the problem of checking whether an element is present as a member of a set. Our elements do not have any special mathematical properties and can be arbitrary, so storing some information about them is unavoidable.

The straightforward approach to this problem is storing the elements or their fingerprints in a collection and then searching the collection to see whether the elements are present or not.

For n -element collections, simple tree structures such as Binary search trees allow for average insertion and retrieval complexity in $O(\log n)$ per element,

while Hashtable based data structures approach $O(1)$. Nevertheless, the space requirement is $O(n)$, so the memory requirements increase linearly as new, unseen data points arrive.

A method first described in [2], the Bloom filter, allows for a significantly reduced memory footprint for the creation of a data structure, at the cost of possible false positives. That is, an element which has not been inserted, can be deemed present in the set with a non-zero probability. The Bloom filter consists of a zero-initialized m -bit array and k hash functions. Each of the hash functions takes the set element and hashes it to a number between 0 and m .

During the **insertion** into the Bloom filter, the element is hashed by each of the hash functions and the bits at the corresponding positions in the bit array are set to 1.

During the **retrieval**, the element is hashed in the same way and the bit positions are examined. If any of them is 0, the element has certainly not been inserted into the array.

Bloom filter can trade off the memory requirements against the false positive rate. The rate is approximately 1 % for a Bloom filter which uses 10 bits per element.

The main drawback is that the Bloom filter does not allow for resizing and that the parameters need to be known in advance. An extension, the Scalable Bloom filter ([1]), allows for indefinitely growable approximate membership structure. First, a single Bloom filter is created. As elements are added and the false positive rate raises above a specified threshold, another larger filter is allocated and new elements are inserted into it. This procedure is repeated as required. Membership is then checked in every Bloom filter in sequence.

Many other data structures for approximate membership testing have been devised over the years with reduced memory requirements, better cache locality or throughput. Unfortunately, all of them seem to have properties which disqualify them for the use case at hand.

For example, the Cuckoo filter ([4]), which uses Cuckoo hashing, is more efficient in terms of space required and exhibits good cache locality, but resizing requires rehashing all the elements which have already been inserted.

The XOR filter ([5]) and Ribbon filter ([3]) are even better in terms of memory requirements, but do not support dynamic insertion and require a distinct build step before they can be used.

3 Blooming Onion

It is written in the Rust³, which is a modern programming language, designed with performance and safety in mind.

The program uses the Growable Bloom filter⁴ library, which implements the Scalable Bloom filter data structure. The Scalable Bloom filter. The structure is initialized with the false positive rate set to 1 %.

³ <https://www.rust.org>

⁴ <https://crates.io/crates/growable-bloom-filter>

Table 1: Susanne corpus

	runtime	max RSS
Blooming Onion	1.94 s	3608 kB
Onion	1.71 s	30616 kB

Table 2: JSI Newsfeed

	runtime	max RSS
Blooming Onion	720.6 s	271.6 MB
Onion	491.3 s	2367.2 MB

Only the most essential features have been implemented at this point and the program serves as a proof of concept. The whole implementation fits into 160 lines of code.

4 Evaluation

Blooming Onion was evaluated against Onion on two datasets:

1. Susanne corpus, repeated 20 times (100 MB, 190 k lines, 97.5 % duplicate, see Table 1)
2. 7 days of the JSI Newsfeed Corpus (13 GB, 876 k lines, 64 % duplicate, see Table 2)

The time required for the deduplication and the maximal resident set size (RSS) have been measured.

While Blooming Onion is about 25 % slower, it uses only 10 % of the memory compared to Onion. The slowdown can be attributed to two major causes. Scalable Bloom filter has worse cache-related behavior compared to the hashtable used by Onion. The backing bits of the filter are scattered around in memory, and therefore require multiple random accesses. This problem could be improved by using a different data structure, perhaps some type of Quotient filter, which orients the accesses for a single elements into a smaller memory are. No implementation of such data structure seems to be available. Of interest could be the fact that the evaluation has been carried out on a server with slow DDR3 memory. A cursory check on a modern laptop with a smaller amount of faster DDR4 memory swaps the order of performance and Onion is slower than Blooming Onion.

The second reason is that Onion is written in a highly optimized way, which avoids many copies of the data at the expense of readability, while Blooming

Onion aims to be simple and readable, and the input text is being copied multiple times. This can be explored in a future version of Blooming Onion.

5 Conclusion

The problem of text deduplication and a current approach we use has been described. The Blooming Onion deduplicator was presented and compared against Onion. Blooming onion is approximately 25 % slower, but only requires 10 % of the compared to Onion. With the proposed improvements, Blooming Onion could be both faster and use more memory compared to Onion.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-CLARIAH-CZ project LM2018101.

References

1. Almeida, P.S., Baquero, C., Preguiça, N., Hutchison, D.: Scalable bloom filters. *Information Processing Letters* **101**(6), 255–261 (2007)
2. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM* **13**(7), 422–426 (1970)
3. Dillinger, P.C., Walzer, S.: Ribbon filter: practically smaller than bloom and xor. *arXiv preprint arXiv:2103.02515* (2021)
4. Fan, B., Andersen, D.G., Kaminsky, M., Mitzenmacher, M.D.: Cuckoo filter: Practically better than bloom. In: *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*. pp. 75–88 (2014)
5. Graf, T.M., Lemire, D.: Xor filters: Faster and smaller than bloom and cuckoo filters. *Journal of Experimental Algorithmics (JEA)* **25**, 1–16 (2020)
6. Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., Suchomel, V.: The sketch engine: ten years on. *Lexicography* **1**(1), 7–36 (2014)
7. Pomikálek, J.: Removing boilerplate and duplicate content from web corpora. Ph.D. thesis, Masaryk university, Faculty of informatics, Brno, Czech Republic (2011)

CompAn – A Tool for Quantitative Comparison of Corpus Annotation

Vlasta Ohlídalová

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
akai@mail.muni.cz

Lexical Computing, Brno, Czech Republic
vlasta.ohlidalova@sketchengine.eu

Abstract. The paper is focused on automatic morphological annotation and its evaluation. The most common evaluation method is described as well as its main issues. Then, based on the theoretical part, a tool for quantitative comparison of corpus annotation (CompAn) is briefly introduced as an alternative to the traditional annotation evaluation based on gold standard corpora.

Keywords: tagging evaluation, gold standard, automatic POS tagging, inter-annotator agreement

1 Introduction

POS tagging is one of the most well researched areas of NLP: the first corpus to be automatically annotated was Brown and it was tagged with the TAGGIT tagger in 1971 [2]. It is also certainly one of the most widely used NLP techniques (both as the first step for developing other tools – such as syntactic analysis – and during linguistic research itself). The accuracy of taggers has been reported at 95–97%, depending on the language and many other variables. Yet similar results have been encountered since the end of the last century [1]. Does this mean that it is sufficient? And perhaps more importantly, is this a good numerical objective indicator of the success of the tools?

In this article I will discuss both of these questions. Based on the theoretical background and issues of the automatic evaluation, which is currently used the most, a tool assisting with the tagging evaluation will be introduced.

2 Is it not good enough yet?

Even using a very simple idea and implementation, the results of POS tagging are quite good (especially when compared to other areas of natural language processing); it currently achieves a success rate just a few percents below 100% – which is also very similar to the level achieved by annotator agreement in the

manually tagged gold standards. Yet all of us who work with corpora know that obvious errors are still around.

And considering the assumed 97% accuracy the automatic taggers achieve, we encounter these errors surprisingly often.

So what is the reason why the real results are often far below the proclaimed ones, and why the accuracy of 97% might still not be enough?

1. The taggers accuracy is calculated from all tokens in the corpus, including, for example, punctuation, on which the success rate can very easily be close to 100%. Moreover, we need to keep in mind how frequent punctuation is: for example, in the English enTenTen15 [3] corpus five of the most frequent tokens are punctuation tokens (comma and period in the second and third places respectively), in the Czech csTenTen17 [3] corpus there are 6 of them in the top 20 (comma and period taking the first two places).
2. The accuracy rate will vary considerably depending on what texts we process. Perhaps the most important problem in this area is the fact that the tagger is usually evaluated on the same type of text it was trained on (although of course on different parts of that text). Thus, it is clear that when such a tagger is run on a different type of text, especially data with a lot of noise such as social network discussions, the resulting percentages may be quite different. The difference between the accuracy of several frequently used taggers on a corpus containing newspaper texts versus a corpus generated from the Web has been addressed throughout the work *Evaluation of POS Tagging for Web as Corpus* by Eugenie Giesbrecht [1]. As expected, all three taggers performed worse on the Web corpus, on average by about 2 percent.
3. The accuracy will also vary considerably depending on the specific genre of the text. In the aforementioned work, an evaluation of accuracy based on genres is also found. The percentages here vary by around 10% – from 88% to 98% (accuracy on each text and its genre is in detail described in Table 1).
4. If we want to build other tools on top of the tagger results, we are often not interested in the accuracy on token level, but rather accuracy on whole sentences (because even one incorrectly annotated token might confuse the tools working with the output). If we take a tagger success rate of 97% and the average sentence length according to the Brown corpus – which is 20 words – the probability of having an error in a sentence is close to 50% (precisely 45.6%). Looking on the issue from the other side, to achieve 95% correctness at the sentence level, we would need an accuracy of 99.6% at the token level – and this is perhaps the number which shows the best how far from it we are.

3 POS tagging evaluation

Evaluation can be theoretically run in many ways (automatic versus manual, formative versus summative, intrinsic versus extrinsic), but in reality, it is

Table 1: Statistics of TreeTagger POS tagging accuracy on various texts in the corpus DeWaC by their genres [1].

genre	overall accuracy
child infections (report)	98.25%
political speech (labor union)	97.52%
job market news	97.46%
news report (school district)	97.10%
scientific news/medicine	96.88%
history (Gold War) report	96.67%
story about Holy Paul	95.42%
biological exposition	94.23%
movie description	93.89%
IT news/Cebit	93.69%
news report (Archbishop)	91.97%
information about a conference	90.98%
Rolling Stones tour (forum)	88.01%

usually reported by comparing the tagger results to a gold standard. That is, the tagger is trained on a part of the manually tagged text and evaluated on another part of the same text (on a part which was not seen before by the tool). Success rate is then reported using accuracy.

Using this method, we might run into the following problems:

- As mentioned above, the genre and type of text plays a role in the final result. So we can assume that whenever a tagger is used in practice and the corpus is not very similar to the one the tagger was trained on, the results will differ. However, this is something which is not recognized at all in the result of this evaluation method.
- Since it is cheap to compare results of a tagger against a gold standard, the comparison can be run as many times as it takes to get the number you are happy with. The focus might therefore easily switch from actually improving the tool to having a number to publish.
- The correctness of the gold standard.

4 Gold standard

The problem with gold standard is that it is considered a fundamental truth and there is no mechanism to deal with the possibility of incorrect annotation. We assume that the labels are always right and never question it, because it is needed both for training a tagger and evaluating their results. A nice example of what inconsistent tagging (for which the Penn Treebank has been known) will ultimately produce is given by Manning [5]. In this paper, 100 mistakes made by the tagger were studied and categorized according to what caused the errors. Of the seven categories, 28% of the errors fell into the

category of “inconsistent/non-existent standard” and another 15.5% even into the “wrong gold standard” category. Together, these accounted for almost half of the errors (43.5% in total). In practice, we see both situations: inconsistency among annotators and mistakes in the gold standard.

4.1 Inter-annotator agreement

A huge problem in many NLP tasks is the (dis)agreement of the linguists themselves. And although we know that this is not as common in POS tagging as in other NLP tasks (especially those involving semantics), the problem exists here too, and given the usually high accuracy of POS tagging, it is important to address it; when we are at 95–97% accuracy, disagreement in 1% of all cases is still a lot and it might make someone wonder how reliable the measured accuracy actually is.

4.2 Incorrect annotation

In addition to disagreements, the ambiguity of the language might also lead to entirely incorrect annotations in the gold standard. A thorough examination of the manually annotated DESAM corpus [6] shows many errors too. For example, the following CQL query run on DESAM returns 13 sentences; of which in 10 cases adjectives are incorrectly annotated as nouns:

```
1:[tag="k1.*" & lemma="[:lower:]].*ý"] 2:[tag="k1.*" &
lemma="[:lower:]].*" & 1.c=2.c within < s/>
```

hlediska lze pochopit pohnutky pozůstalých a	známých obětí	surových a zbytečných vražd , například
k1gNnSc2 k6eAd1 k5eAaPmF k1gFnPc4 k1gMnPc2 k8xC	k1gMnPc2 k1gFnPc2	k2eAgFnPc2d1 k8xC k2eAgFnPc2d1 k1gFnPc2 k1x k6eAd1
kou zradu " .</s><s> Dnes se 44 tisíc	mladých občanů	naši republiky vojenské službě , například
ld1 k1gFnSc4 k1x k6eAd1 k3xPyF4 k4xCglnPc2	k1gMnPc2 k1gMnPc2	k3xOp1gFnSc2 k1gFnSc2 k2eAgFnSc3d1 k1gFnSc3 k5eAaImAglnS
ledu .</s><s> Vážný důvod , návštěva těžce	nemocného otce	doma v Rusku , byl přesto s
iSc2 k2eAglnSc1d1 k1glnSc1 k1x k1gFnSc1 k6eAd1	k1gMnSc2 k1gMnSc2	k6eAd1 k7c6 k1gNnSc6 k1x k5eAaImAglnS k8xC k2eAglnSc1d1
run .</s><s> " Ve výpovědích se oba	obvinění příslušníci	v tomto bodě rozcházejí , " například
nPc2 k1x k7c6 k1gFnPc6 k3xPyF4 k4xCgMnPc1	k1gMnPc1 k1gMnPc1	k7c6 k3xDglnSc6 k1glnSc6 k5eAaImP3nP k1x k5eAaImAglnS

Fig. 1: A few lines showing incorrectly annotated tokens in DESAM.

5 A tool for quantitative comparison of corpus annotation

The previous sections have described problems in automatic evaluation of morphological annotation that can – and often do – lead to inaccurate results. For this reason, the CompAn tool (the name comes from “compare annotations”) has been created. Although the tool does not evaluate the quality of the morphological annotation, it compares the annotation of any attribute (at the

	Freq	rftagger	rftagger_synt	Conc	Conc
1	112	k1gInSc1	k4		
2	69	k1gMnSc1	k1gInSc1		
3	59	kF	k4		
4	45	k1gInSc4	k4		
5	39	k7c6	k7c4		

Fig. 2: The example output of the tool when comparing attribute value (tags in this case)

moment it can be used for token, lemma or tag) in a single corpus processed by two different tools (such as a tokenizer, part-of-speech tagger or lemmatizer).

In practice, this means that the tool will list the most frequent differences in the annotations of the two tools, either by attribute value (i.e. which values were most often interchanged) or by word (i.e. which words were most often annotated differently). Examples of how the results are displayed in the interface are shown in Figure 2 and Figure 3.

	Freq	Word	rftagger	rftagger_synt	Conc	Conc
1	26	v	k7c6	k7c4		
2	14	ponděli	k1gNnSc6	k1gNnSc4		
3	8	na	k7c6	k7c4		
4	7	top	kA	k5nSp2		
5	7	to	k3gNnSc1	k3gNnSc4		

Fig. 3: The example output of the tool when comparing words

Thus, the tool is not designed to produce one particular number that can be used as a universal indicator of annotation quality, but rather to assist in the manual comparison of two tools. The results are first pre-calculated and cached, so that they can be retrieved instantly to be searched by different criteria.

The tool was developed as a RiotJS web application with a Python backend that uses corpora indexed by Manatee which is part of the (No)Sketch Engine corpus management suite [4,7]. Any indexed corpus can be instantly loaded and evaluated by CompAn.

6 Conclusions

In this paper CompAn is presented, an online tool for comparing annotations between two corpora. The main motivation behind the tool is comparison of part-of-speech annotation, lemmatization or tokenization, but it can be easily generalized for other purposes as well.

References

1. Giesbrecht, E.: Evaluation of POS Tagging for Web as Corpus. Master's thesis
2. Greene, B.B., Rubin, G.M.: Automated grammatical tagging of English. Dept. of Linguistics, Brown University (1971)
3. Jakubíček, M., Kilgarrieff, A., Kovář, V., Rychlý, P., Suchomel, V.: The TenTen corpus family. In: 7th International corpus linguistics conference CL. pp. 125–127. Lancaster University (2013)
4. Kilgarrieff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., Suchomel, V.: The Sketch Engine: ten years on. *Lexicography* pp. 7–36 (2014)
5. Manning, C.: Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? vol. 6608, pp. 171–189 (05 2011). https://doi.org/10.1007/978-3-642-19400-9_14
6. Pala, K., Rychlý, P., Smrž, P.: DESAM – Annotated Corpus for Czech. In: *Proceedings of SOFSEM '97*. pp. 523–530. Springer-Verlag (1997)
7. Rychlý, P.: Manatee/Bonito-A Modular Corpus Manager. In: *RASLAN*. pp. 65–70 (2007)

Part III

Text Corpora

Piötòst Ché Nient, Mèi Piötòst - A Manually Revised Lombard-Italian Parallel Corpus

Edoardo Signoroni

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
e.signoroni@mail.muni.cz

Abstract. The Lombard language is a Gallo-Italic language spoken in the Northern Italian region of Lombardy and some surrounding areas by 3.5 million native speakers in varied spectrum of bilingual settings and fluency. However, it is currently listed as “definitely endangered” according to UNESCO. Despite some resurging interest in documenting, revitalizing, and using the language, no Natural Language Processing resource was specifically build for Lombard. The only existing Lombard-Italian parallel corpus was created as part of a bigger multilingual project by scraping aligned text from Wikipedia articles. However, we found the resulting corpus to be faulty, due to noise and erroneous alignments. Our work addresses these issues by providing a cleaner, human-revised version of this resource, which could be used as a stepping stone to build future NLP tools, such as a Machine Translation system.

Introduction

Lombard is a regional language¹ spoken in and around the Northern Italian region of Lombardy by about 3.1 million people,² where it exists alongside the official language, Italian, in varying degrees of bilinguality and fluency. It belongs to the Gallo-Romance-Cisalpine group of the Western Romance family of the Indo-European languages, and it is said to have between two and four varieties, the main ones being Western (in the provinces of Varese, Como, Lecco, Sondrio, Milan, Monza, Pavia and Lodi, in addition to Novara and Verbania in Piedmont and Canton Ticino in Switzerland) and Eastern Lombard (in the provinces of Bergamo, Brescia and Northern Cremona). These varieties, even with some phonetic, lexical, and grammatical differences, can

¹ This definition is preferred over the one commonly used today, even by some academics, of *dialeto* (en. “dialect”, following Coseriu’s (1981) [8] definition of so-called “primary dialects”), which is arguably both erroneous and derogatory. [5] As Chambers and Trudgill [4] state: “a dialect is a substandard, low status, often rustic form of language, generally associated with the peasantry, the working class, or other groups lacking in prestige”.

² To these figures, which report numbers just from Lombardy, one must add speakers in neighboring regions and from Switzerland. Data according to Istituto Nazionale di Statistica (ISTAT) from 2015 <https://www.istat.it/it/archivio/207961>

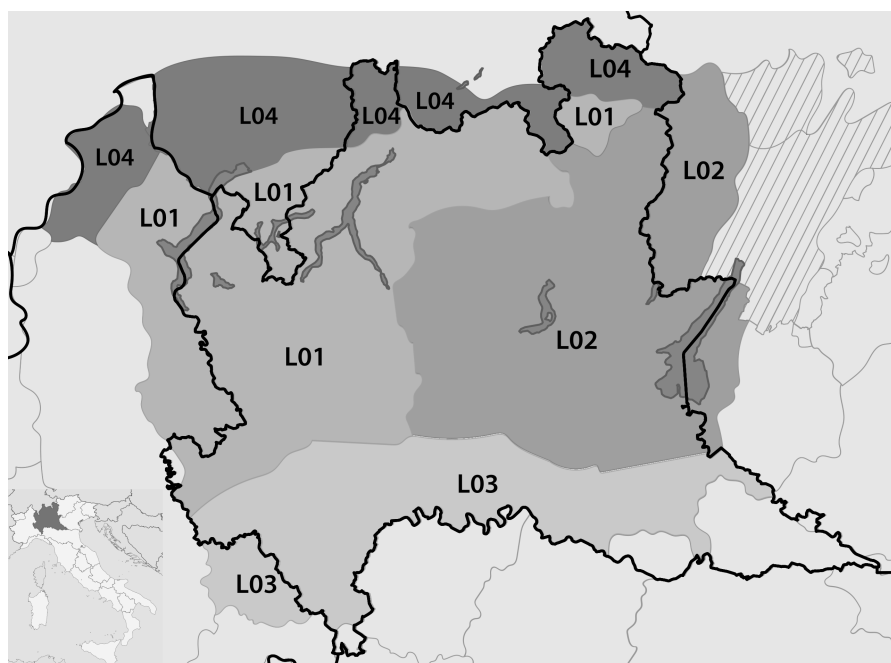


Fig. 1: Map showing the geographical distribution of the Lombard language and its varieties according to a fourfold subdivision. L01 denotes Western Lombard, L02 Eastern Lombard, L03 Southern Lombard, and L04 Alpine Lombard. Image retrieved from https://commons.wikimedia.org/wiki/File:Mappa_Dialetti_lombardi.svg

be loosely considered to be one language, since they are mutually intelligible. [7,2,11,6] At the present day, the language is mostly used in oral conversation and no unified orthography exists, with different approaches ranging from phonetic/phonemic, to historical or etymological ones. One of these is the proposal by Brasca (2011). [3] Figure 1 shows the location of Lombard and its variants in Northern Italy.

Despite the relatively large amount of speakers, and featuring literature and cultural activities in different forms, the current status of Lombard is of concern due to a plethora of reasons, being them historical, social, political, or legislative. Discussing these issues, most of which are complex and controversial (at least for an Italian audience), lies outside the scope of this paper.³ UNESCO [14] lists Lombard as a “Definitely endangered” language⁴ According to other

³ To more in-dept discussion on this topic, see the works referenced in the bibliography.

⁴ A language that “is no longer being learned as the mother tongue by children in the home. The youngest speakers are thus of the parental generation. At this stage, parents may still speak their language to their children, but their children do not typically respond in the language.”

metrics, such as EGIDS (Extended Graded Intergenerational Disruption Scale) [10], Lombard is between grades 6b “threatened” and “moribund”. However, some interest in Lombard, and other regional languages of Italy, is resurging with some cultural and multimedia production, academic research, and even social network and Wikipedia⁵ pages. Moreover, in 2016 a regional law⁶ was passed for the protection and promotion of Lombard.

If a thorough effort towards this goal has to be made in the present day, Natural Language Processing (NLP) resources must to be developed. Among such technologies, a Machine Translation (MT) system and its foundational basis, a parallel corpus, can surely be beneficial to the preservation of the language. The only existing Lombard-Italian parallel corpus was created as part of a bigger multilingual project by scraping aligned text from Wikipedia articles. However, we found this corpus to be faulty, due to the widespread presence of noise and erroneous alignments. This work addresses this issue by providing a cleaner, human annotated version of this resource on top of which build NLP tools, such as a Machine Translation system.

This paper is structured as follows: Section 1 briefly surveys previous work on Lombard; Section 2 relates the methodology of this work and describes the resulting corpus; Section 3 discusses its limitations and outlines some future work to address them; Section 4 presents our conclusions.

1 Related Work

Regarding NLP work on Lombard, not much has been done. Glottolog⁷ lists published research on Lombard variants from the 19th century to 2021. Most of the current research has focused on sociolinguistics and revitalization. Some systematic documentation of the language, or its variants, has been carried out in the form of lexical atlases, such as the one by the Fondazione Civiltà Bresciana.⁸

While books in Lombard (most likely one of its variants) can be found in physical circulation, the digitalization of textual sources is lacking, with not even a full text of the Bible⁹ freely obtainable online, the only text available being dictionaries and parts of the Gospel.

As far as concrete NLP resources are concerned, Lombard monolingual corpora are available only as part of larger projects with Wikipedia dumps [15], such as W2C [12], and Deltacorpora [13]. To our knowledge, no monolingual corpus has been built specifically for Lombard.

⁵ https://lmo.wikipedia.org/wiki/Pagina_principala

⁶ Regional Law no. 130/2016

⁷ <https://glottolog.org/resource/languoid/id/lomb1257>

⁸ <https://www.civiltabresciana.it/pubblicazioni/atlantelessicale.html>

⁹ The Bible is usually the go-to source for unresourced languages, since it is the most widely translated book in the world and comes with the advantage of having a built-in “gold” alignment in the form of verses.

Posso capire perché pensò a me. A gh'è nissün che 'l pensa a mì.

Fig. 2: An example of a wrong alignment. The translation of the sentences are as follows: it. *"I can think why he/she thought about me."* Imo. *"There is no one who thinks about me."*

With regards to parallel corpora, the only readily available one is the parallel corpus in OPUS. [17]¹⁰ It consists of the Lombard-Italian section of the WikiMatrix corpus [16] automatically created by mining parallel sentences from Wikipedia articles through multilingual sentence embedding similarity. [1] This resource was revealed to be very noisy and plagued by errors after our preliminary evaluation of a sample of the proposed sentence pairs.

2 Methodology

2.1 Preliminary evaluation

Our work started with evaluating a sample of the corpus available on OPUS. We manually analyzed 500 sentence pairs and determined that 157 were incorrect. This amounts to 31.4% of the sample being judged either as errors or noise. The most common instances of these were duplication of the sentence on both sides, a fully or partially incorrect alignment, or similar sentences or context that were nonetheless incorrect translations. In some cases, these were loose paraphrases or summarizations of the Italian text. Where these could be easily fixed, that is if the extent of the error was roughly under half of the overall length of sentence, we modified the Italian sentence to match the Lombard one. We did not modify the Lombard side of the alignments to avoid the injection of further noise in the data, e.g. through subjective spellings or orthographical choices.

It is relevant to note that some of the removed examples contained well formed sentences on the Lombard side. Recovering and complementing these phrases is left to future work, but it signals that a bigger amount of data may be available to be exploited. Figure 2 gives an example of an incorrect alignment to be removed.

2.2 Manual annotation

We then moved on to manually revise the whole parallel corpus, which amounts to 10.533 sentence pairs. These were divided among five different annotators, all native bilingual speakers of Italian and Lombard, more precisely the Brescian variety of Eastern Lombard.¹¹ While it can be argued that this annotator group may bias the results, we maintain that this risk, while present, is very low for the task we carried out. Our reasoning is the following.

¹⁰ <https://opus.nlpl.eu/>

¹¹ The author of this paper is also among the annotators.

Total	Correct	Removed	Modified
10.533	4915 46.67%	5227 49.62%	391 3.71%

Table 1: Number of correct, removed, and modified alignments against the starting total.

First, the annotators did not provide or choose any kind of data for the corpus; their task was to judge the correctness of the alignments, which were independently generated by an automated method. Moreover, as stated in Section 2.1, even in the instances in which the alignments were manually corrected, instead of being removed all-together, only the Italian side was modified in order to avoid the insertion of subjective forms and orthography in the text.

Second, similar work [9] found that relatively simple annotation tasks such as evaluating the correctness of a sentence alignment can be carried out effectively even by annotators with little or no proficiency of the languages under scrutiny. The annotators were all native bilingual speakers of Italian and a Lombard variety. Recall from the Introduction that the varieties of Lombard are to a great extent mutually intelligible, thus being proficient in one of them should suffice for this annotation task.

In our manual revision, we removed 5227 pairs, or 49.62% of all the alignments, and modified a further 391, the 3.71% of the total. The pairs deemed to be already correct were 4915, amounting to 46.67% of the total. Thus, the final corpus has a total size of 5306 sentence pairs. Table 1 gives the numbers of correct, removed, and modified pairs against the original size of the corpus.

2.3 Corpus

After the revision the corpus has 5306 sentence pairs, the 50.37% of the initial 10.533. The Lombard side has 122.550 tokens,¹² the Italian one has 113.385, for a total of 236.264 tokens. The average sentence length in tokens is 23.10 for Lombard and 21.37 for Italian. Table 2 summarizes these statistics.

N. of pairs	N. of words		Avg. sentence length	
	LMO	IT	LMO	IT
5306	122.550	113.385	23.10	21.37

Table 2: Some figures about the revised corpus: the total number of sentence pairs, the number of whitespace-separated tokens, and the average sentence length for each side.

¹² Here a token is intended a string separated by whitespace.

3 Limitations and Future work

Despite being cleaner, the corpus is definitely small, both in scale and scope. It will have to be expanded with data from other domains and sources to be more impactful for the training of an MT system which can generalize well across different domains.

Another limitation of the corpus, which is however inherent in Lombard text, is the lack of standardisation in orthography. As you may recall from the Introduction, at the present moment, there is no generally accepted orthography for Lombard and its varieties. This is reflected in the Lombard Wikipedia, where pages are written in one of the proposed orthographies and varieties, which is signalled by a disclaimer on the top of the page. This is an issue if this corpus is used in the training of a NLP system, since words with the same meaning and contexts of use will be present in different forms, with lower frequencies and thus, with worse representations.

Future work will aim to solve these issues from a NLP perspective. Applying Optical Character Recognition tools to existing text may be worthy of investigation as a way to augment the size of the corpus. A tool to convert text to a uniform orthography could be devised leveraging existing dictionaries and standardisation proposals.

4 Conclusions

This work focused on Lombard, a Gallo-Romance regional language spoken in and around the Northern Italian region of Lombardy. Despite having more than 3.5 million speakers, no NLP resource has ever been created specifically for this language, with most of the research concentrating on documentation and sociolinguistics issues.

This work thus focused on providing a first foundational NLP resource for Lombard, a manually revised parallel corpus starting from the only Lombard-Italian resource available on-line. This corpus was created automatically mining parallel text from Wikipedia, and was found to be noisy. Thanks to the manual revision of five annotators, all bilingual native speakers of both Italian and Lombard, we obtained a cleaner corpus, which is available on GitHub.¹³

While being small¹⁴, this is a first step towards providing NLP tools to users of the Lombard language, hopefully securing its precarious position in the diverse and complex linguistic landscape of Italy.

Acknowledgments The author wishes to thank the volunteer annotators, whose help was vital. We also want to thank the reviewers for their useful comments. The author's work is partially supported by the Internal Grant Agency of Masaryk University, project MUNI/IGA/1334/2021, and by Lexical Computing.

¹³ <https://github.com/edoardosignoroni/piotost>

¹⁴ That is

References

1. Artetxe, M., Schwenk, H.: Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics* 7, 597–610 (nov 2019). https://doi.org/10.1162/tacl_a_00288, https://doi.org/10.1162/2Ftacl_a_00288
2. Bonfadini, G.: lombardi, dialetti. In: Eds., T. (ed.) *Enciclopedia dell'italiano*. Treccani, online at [https://www.treccani.it/enciclopedia/dialetti-lombardi_\(Enciclopedia-dell%27Italiano\)/#Studi](https://www.treccani.it/enciclopedia/dialetti-lombardi_(Enciclopedia-dell%27Italiano)/#Studi) (2010)
3. Brasca, L.: *Scriver Lombard*. Menaresta, Monza, 1st adj. edn. (2011)
4. Chambers, J.K., Trudgill, P.: *Dialectology*. Cambridge Textbooks in Linguistics, Cambridge University Press, 2 edn. (1998). <https://doi.org/10.1017/CBO9780511805103>
5. Coluzzi, P.: The new speakers of lombard. *Multilingua* 38(2), 187–211 (2019). <https://doi.org/doi:10.1515/multi-2018-0017>
6. Coluzzi, P., Brasca, L., Scuri, S.: Revitalizing contested languages: The case of lombard. In: Tamburelli, M., Tosco, M. (eds.) *Contested Languages: The hidden multilingualism of Europe*, chap. 9, pp. 163–182. John Benjamins, Amsterdam (2021)
7. Coluzzi, P., Brasca, L., Trizzino, M., Scuri, S.: Language planning for italian regional languages: the case of lombard and sicilian. In: Stern, D., Nomachi, M., Belić, B. (eds.) *Linguistic Regionalism in Eastern Europe and Beyond: Minority, Regional and Literary Microlanguages*, pp. 274–298. Peter Lang, Frankfurt am Main (2018)
8. Coseriu, E.: *Los conceptos de dialecto, nivel y estilo de lengua y el sentido propio de la dialectología* (1981)
9. Kreutzer, J., Caswell, I., Wang, L., Wahab, A., van Esch, D., Ulzii-Orshikh, N., Tapo, A., Subramani, N., Sokolov, A., Sikasote, C., Setyawan, M., Sarin, S., Samb, S., Sagot, B., Rivera, C., Rios, A., Papadimitriou, I., Osei, S., Suarez, P.O., Orife, I., Ogueji, K., Rubungo, A.N., Nguyen, T.Q., Müller, M., Müller, A., Muhammad, S.H., Muhammad, N., Mnyakeni, A., Mirzakhlov, J., Matangira, T., Leong, C., Lawson, N., Kudugunta, S., Jernite, Y., Jenny, M., Firat, O., Dossou, B.F.P., Dlamini, S., de Silva, N., Ballı, S.Ç., Biderman, S., Battisti, A., Baruwa, A., Bapna, A., Baljekar, P., Azime, I.A., Awokoya, A., Ataman, D., Ahia, O., Ahia, O., Agrawal, S., Adeyemi, M.: Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics* 10, 50–72 (2022). https://doi.org/10.1162/tacl_a_00447, https://doi.org/10.1162/2Ftacl_a_00447
10. Lewis, M.P., Simons, G.F.: Assessing endangerment: Expanding fishmans's gids (2010), <https://www.lingv.ro/RRL%20202010%20art01Lewis.pdf>
11. Loporcaro, M.: *Profilo Linguistico dei Dialetti Italiani*. Manuali Laterza, Editori Laterza, Bari, 1st edn. (2009)
12. Majliš, M.: W2C – web to corpus – corpora (2011), <http://hdl.handle.net/11858/00-097C-0000-0022-6133-9>, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University
13. Mareček, D., Yu, Z., Zeman, D., Žabokrtský, Z.: Deltacorpora 1.1 (2016), <http://hdl.handle.net/11234/1-1743>, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University
14. Moseley, C., Nicholas, A.: *Atlas of the World's Languages in Danger*, Memory of Peoples, vol. 19. UNESCO, Paris, 3rd edn. (2010)
15. Rosa, R.: Plaintext wikipedia dump 2018 (2018), <http://hdl.handle.net/11234/1-2735>, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University

16. Schwenk, H., Chaudhary, V., Sun, S., Gong, H., Guzmán, F.: Wikimatrix: Mining 135m parallel sentences in 1620 language pairs from wikipedia. CoRR **abs/1907.05791** (2019), <http://arxiv.org/abs/1907.05791>
17. Tiedemann, J.: Parallel data, tools and interfaces in OPUS. In: Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12). pp. 2214–2218. European Language Resources Association (ELRA), Istanbul, Turkey (May 2012), http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf

Aranea Go Middle East: Persicum

Vladimír Benko^{1,2} 

¹ Slovak Academy of Sciences, L. Štúr Institute of Linguistics
Panská 26, 811 011 Bratislava, Slovakia
<http://juls.savba.sk/~vladob>
vladimir.benko@juls.savba.sk

² Comenius University Science Park
UNESCO Chair in Plurilingual and Multicultural Communication
Ilkovičova 8, 814 04 Bratislava, Slovakia

Abstract. Our paper introduces the creation and annotation of *Araneum Persicum*, a new Persian web-crawled corpus. Some problems encountered during the process of filtration and annotation are shown, and an ensemble approach adopted for lemmatization and morphosyntactic annotation is introduced. It is also argued that Romanization can be helpful in developing corpora for languages not based on Latin script.

Keywords: Web-crawled corpus, Persian language, Ensemble tagging

1 Introduction

The Aranea Project³ [2] aimed at the creation and annotation of a family of web-crawled corpora for languages taught at Slovak Universities has reached the point where most “common” languages have been covered already, and their total count has approached the two-dozen landmark. For various reasons, new languages are still being added to our collection, even if there is no chance that they would ever be taught in Slovakia. The Persian language, also referred to as Farsi⁴, belongs to this category as well.

Our attempt to build a Persian corpus has been initialized by our Prague colleagues working on a Persian to Czech dictionary [15] who need a reliable source of lexical evidence on contemporary Persian language, as well as our desire to make use of our experience and tools developed in the framework of our Aranea Project to process a language using a right-to-left script.

2 The Persian Language

Persian belongs to the Indo-Iranian subgroup of Indo-European languages with at least 70 million speakers⁵. If all its varieties are considered, the language

³ <http://aranea.juls.savba.sk/guest/>

⁴ <http://www.iranian.com/Features/Dec97/Persian/>

⁵ https://en.wikipedia.org/wiki/Persian_language

has official status in Iran, Afghanistan, and Tajikistan, as well as in several neighboring countries. Having a long history of writing, the modern Persian uses a modified Arabic script in Iran and Afghanistan, and a modified Cyrillic script in Tajikistan.

2.1 The Persian Script

The main obstacle in any attempt to grasp the Persian script is the fact that the shape of (almost) any grapheme can have as many as four different forms depending on its position within a word (initial, medial, final, and isolated, respectively). To ease this “mental burden”, we decided to supplement each corpus token (word form, lemma, etc.) by its respective Romanized variant, adopting the UN 2012⁶ transliteration system. The main advantage of this system for use in our environment is that all transliterated graphemes are directly accessible via Czech and Slovak keyboard, with the only exception being the “ā” character (representing the Persian “ا”) that has been substituted by an “á”.

In comparison to Arabic, the Persian script contains four additional graphemes representing phonemes not present in Arabic (پ, چ, ژ, گ, transliterated as “p”, “č”, “ž”, and “g”, respectively), and two graphemes (ی, ک, i.e., “y” and “k”) that have slightly different shapes and their own Unicode code points – this fact can be conveniently used in secondary language filtration.

The real-world Persian texts on the web, however, also contain certain amount of words with Arabic spelling (mostly proper names and Quran-related lexical items), loanwords from other Indo-Iranian languages preserving the original orthography, nonstandard use of diacritics denoting vowels, etc., so some sort of normalization is suggested before a text can be processed by a NLP tool.

2.2 Persian Morphology

I must admit that I was only able to “plunge” into those issues in this area that generated some problems during lemmatization and PoS tagging of the corpus data.

Unlike in most other languages within the Aranea family, the basic form of a Persian verb is not represented by its infinitive, but rather by two stems (present and past, respectively). This has a rather negative influence on lemmatizers that are in such a case typically not able to guess valid lemmas for the out-of-vocabulary (OOV) lexical items.

Another peculiar feature of the Persian morphology is that certain affixes can be written either together with the stem, separated by a “half-space” (“zero-width non-joiner” character U+200c, having a special hotkey combination on a Persian keyboard), or even by a standard space. A corpus designer therefore has to make a decision about what data should be sent to the tagger (i.e., the original, half-space-normalized or even space-normalized).

⁶ http://www.eki.ee/wgrs/rom1_fa.pdf

3 Persian Language Resources and NLP Tools

Even a simple “Google research” reveals lots of projects devoted to the processing of the Persian language. On the other hand, resources that would be readily available to those who would like to compile their own Persian corpora themselves are not so numerous. In general, at least tools for lemmatization and/or PoS tagging are needed. To speed up the creation of the initial (beta) version of our Persian corpus, we decided to make use of only those tools that have been already engaged in the processing of other corpora of the Aranea family.

Persian Treebanks. The obvious place to look for (syntactically) annotated corpora is the *Universal Dependencies Portal*⁷. We can find two Persian items there. The larger is the *Persian Dependency Treebank (PerDT)* [8] containing approximately 500 K tokens, while the considerably smaller *Seraji* [11] based on the *Uppsala Persian Treebank* has 152 K tokens. Besides its size and genre coverage of the latter (it contains news only), the other issue is its “incomplete” lemmatization – lemmas for many lexical items are simply set to an “_” character. If used for training, this error is further propagated to the tagged data.

TreeTagger [10]. Despite its age, this tool is still being used by many corpus projects, including that of ours. There are several reasons for this: Firstly, it is still maintained by its original developer; secondly, there are language models for many different languages; thirdly, it is stable even if applied on very large (many Gigaword) corpora; and lastly, it is very fast – especially in comparison to newer tools based on, say, a neural network.

On the other hand, the quality of its output is not as high as that of newer taggers, especially if applied to a language with rich inflectional morphology and corresponding fine-grained tagset [4]. The *TreeTagger* performs guessing of PoS tags for OOV lexical items, yet it does not attempt to guess the lemma in such a situation.

The Persian language model for *TreeTagger* has been created by means of the *PerDT* data which means that it is the tool with the largest coverage of Persian lexis.

UDPipe [12] is the main tagging tool developed within the *Universal Dependencies Project* [5]. The corresponding language model(s) can be created for all languages where a corresponding treebank exists. As the *UDPipe 3* version is still in development and therefore not released yet, and the *UDPipe 2* is available as a Python prototype (or a web service) only, i.e., not suitable for processing large-scale data [13], for users who want to run the software within their own infrastructure, the oldest *UDPipe 1* is the only option.

Although two different treebanks for Persian are available, for various reasons the only language model available has been trained on the *Seraji Treebank*

⁷ <https://universaldependencies.org/>

resulting into a much lower coverage than that of *TreeTagger*, and also leaving many lexical items without any lemma.

CSTlemma [3]. As its name suggests, this tool does not perform a “complete” tagging, and just generates basic form for each token in the corpus. The respective language model can be trained by a (preferably large) morphological lexicon, and authors provide pretrained models for many languages. The Persian model has been created by means of the MULTEXT-East lexicon [16], and (what is its rather negative feature for our work) does not generate “compatible twin lemmas” for Persian verbs.

4 Corpus Processing

Preparation. The first step in building a new web-crawled corpus is the collection of seed URLs that are needed as one of the inputs for the *SpiderLing* [14] crawler. This used to be fairly easy to perform by the *BootCaT* [1] tool, until Microsoft stopped supporting the free Bing queries via an API some years ago. The tool itself is still operational, yet the current procedure involves a lot of manual “cut and paste” operations, which makes this option clearly rather “suboptimal”. An alternative is provided by the *WebBootCaT* functionality of the *Sketch Engine*⁸ portal (if one owns an account ;-)

The procedure of “harvesting” the URLs involves providing the program with a set of “keywords” used to create n-grams that are being submitted to a search engine. The resulting lists of Internet addresses may be manually edited and used for the subsequent downloading of the actual documents. In our case, however, we did not need to let Sketch Engine to perform the downloading, and just took the list itself. This procedure can be repeated until the required number of URLs has been collected.

According to our experience, the initial list of keywords should consist of words of general semantics, such as high-frequency adverbs. The respective list for our work has been extracted from one of the Persian corpora hosted at the Sketch Engine site: the list of most 1,000 adverbs has been randomly sorted and 5 sets of 12 words have been used. The n-gram length has been set to 3 and all other parameters to the maximal values. The five rounds of harvesting yielded (after deduplication and removing URLs from unwanted domains, such as *instagram.com* and *youtube.com*) approximately 19,500 URLs.

Another user input needed for *SpiderLing* operation are text samples used to create language models for on-the-fly language identification and filtering during the crawling. Samples for Persian, Arabic, and English have been extracted from selected Wikipedia pages in the respective languages.

Crawling and preprocessing. The actual crawling was performed in April 2022 by *SpiderLing 2.0* in 10 parallel threads. After some 36 hours of crawling, approx.

⁸ <https://www.sketchengine.eu/>

96 GB of raw text in a “prevertical” format have been gathered, almost 20 GB out of which have been removed during the initial deduplication targeted at 100% duplicates.

Two main filtering procedures attempted to delete texts being “insufficiently Persian” (counting frequencies of Persian characters) and “too non-Persian” (counting characters not present in the Persian alphabet, yet being based on Arabic script). The respective thresholds in both cases have been set experimentally, removing in total another 21 GB of data. A short analysis of the removed documents revealed that, besides Arabic, most of them were in fact written in the Pashto language that is also spoken both in Iran and Afghanistan.

Tokenization. In the framework of our Aranea Project, the universal tokenizer *Unitok* [6] (with custom parameter files) is used for tokenization. After initial experimentation and somewhat surprisingly, the English parameter file could be used (almost) without modification for Persian – the only change was associated with the treatment of “half-spaces” that had to be considered “letters”, if non-normalized text is to be tokenized.

The tokenization procedure yielded a vertical file of 5.59 Gigatokens. The secondary deduplication procedure (performed by Onion [7]) removed more than 30% of them, retaining the 3.89 Gigatokens in approx. 4.49 M documents.

Ensemble annotation. In Computational Linguistics, the “ensemble” term is used to describe approaches where several tools are utilized to (independently) perform the same operation, assuming that aggregation of their outputs could improve the overall success rate of the whole process. In the framework of morphosyntactic annotation, we can speak about “ensemble tagging” if more than one tagger is available for a particular language – which is also the case of Persian.

If all the tools use the same tagset and they are more than two, the aggregation is usually performed by simple “voting”. In our case, however, we not only do not have three “full-fledged” taggers, and the respective tagsets are not completely compatible. The component tools also do not behave in the same way with respect to OOV lexical items. The actual situation is shown in Table 1.

Table 1: Three ensemble component tools.

Word forms	TreeTagger	UDPipe	CSTlemma
Non-OOVs	PoS tag assigned	PoS tag guessed	n/a
OOVs	PoS tag guessed	PoS tag guessed	n/a
Non-OOVs	Lemma assigned	Lemma guessed	Lemma guessed
OOVs	Lemma marked as OOV	Lemma guessed	Lemma guessed

The aggregation procedure has been therefore designed as follows:

1. Only main word classes (based on the *AUT*⁹ tagset) are considered for aggregation.
2. If PoS can be assigned by means of a regular expression (punctuation, digits, symbols, e-mail addresses, etc.), ignore the information from the taggers.
3. If the respective token was present in the morphological lexicon of TreeTagger, take both lemma and PoS from it.
4. If the token was OOV in TreeTagger and UDPipe guessed a lemma, take both lemma and PoS from it. If lemma guessed by CSTlemma differs, add it as an alternative. If PoS guessed by the TreeTagger differs, add it as an alternative.
5. Otherwise take the lemma from CSTlemma and PoS from TreeTagger and UDPipe (if it differs).

The result of the aggregation process is flagged in a special attribute “ztag”: the respective value consists of two parts separated by a period – the left part denotes assignment of lemma, while the right that of the PoS. The uppercase letters indicate success in the morphological lexicon lookup (in case of TreeTagger), the lowercase letters indicate guessing and the exclamation mark indicates that the respective value differs from that on the left. The actual situation in a 125-Megatoken sample of *Araneum Persicum* is shown in Table 2.

Table 2: Frequency distribution of ztags.

ztag	freq	%	ztag	freq	%
T!c.T!u	313,397	0.25	u!c.t!u	734,753	0.59
T!c.Tu	1,744,755	1.40	u!c.tu	2,098,439	1.68
T!u!c.T!u	141,740	0.11	uc.t!u	2,077,455	1.66
T!u!c.Tu	2,082,343	1.67	uc.tu	3,751,262	3.00
T!uc.T!u	644,554	0.52	c.t!u	1,411,739	1.13
T!uc.Tu	4,477,277	3.58	c.tu	3,026,913	2.42
Tc!u.T!u	783,445	0.63	z!	6,629,046	5.30
Tc!u.Tu	1,242,515	0.99	z#	788,956	0.63
Tc.T!u	694,332	0.56	z\$	910,190	0.73
Tc.Tu	3,817,999	3.05	z@	1,068	0.00
Tu!c.T!u	455,065	0.36	zu	5,987	0.00
Tu!c.Tu	13,464,342	10.77	zv	17,675	0.01
Tuc.T!u	7,834,406	6.27	zw	1,972	0.00
Tuc.Tu	65,848,758	52.68	Total	125,000,383	100.00

Flags starting with the “z” letter indicate lexical items “tagged” by regular expressions. For example, “z!” denotes punctuation, “z#” numbers, and “z\$” symbols (special graphic characters, emoji, etc.).

As it can be seen, most items have a “Tuc.Tu” flag, indicating equal values assigned by all tools, followed by a “Tu!c.Tu”, with CSTlemma assigned a different lemma than TreeTagger and UDPipe.

⁹ http://aranea.juls.savba.sk/aranea_about/aut.html

5 Compilation by the Corpus Manager and Publication

During the development of the corpus, a small sample of the whole data was used with all parallel annotations being available for querying via the *NoSketch Engine* [9] corpus manager. This helped to identify several issues of the processing pipeline, as well as the respective tools themselves. The final beta version of the corpus containing all data, however, contains the aggregated annotations only plus the transliterated versions of both word and lemma attributes. By including these fields into the SIMPLEQUERY directive of the corpus configuration file, it is now possible to conveniently query either in Persian or transliterated versions of the respective attributes. An example of such a query is shown in Fig. 1).

The screenshot shows the Araneum Persicum Beta Minus search interface. At the top, there is a search bar with the query 'brnv' and a magnifying glass icon. To the right of the search bar, it says 'Araneum Persicum Beta Minus (Farsi, 22.10.ter) 125 M'. Below the search bar, there is a status bar indicating 'Query brnv 32 (0.26 per million)' and a progress bar showing 'Page 1 of 2'. The main content area displays a list of search results, each consisting of a source URL on the left and a snippet of text on the right. The results include links to ahangchin.ir, article.mojahedin.org, golestan24.com, military.ir, etood.com, sayarnews.ir, fa.wikidark.org, news.gooya.com, ketabrah.ir, wikipedia.net, digiato.com, parsizi.ir, zendegisalam.ir, power-music.ir, toseeirani.ir, and anthropologyandculture.com. The snippets contain Persian text related to the query 'brnv' (Brno).

Fig. 1: The result of querying “brnv” (“Brno”) in Araneum Persicum Minus.

The Beta versions of the Persian corpus in three sizes have been published at our Aranea Corpus portal recently.

6 Conclusion and further work

Despite the fact that the processing pipeline for Aranea corpora has been tuned and is relatively stable, any new corpus may present additional challenges, let alone in situations, when the developer(s) do not understand the language. The ensemble approach for lemmatization and tagging significantly improved

the quality of annotation. The idea of providing supplementary transliterated attributes turned out to be quite successful and made the tuning of the data much easier.

For the next version of *Aranuem Persicum* we would like to add more component tools to the ensemble, and maybe also try to create own language model for UDPipe based on PerDT.

Acknowledgements This work has been, in part, supported by the Slovak VEGA Grant Agency for Science, Project No. 2/0016/21, as well as by the TAČR Technology Agency of the Czech Republic, Project No. TL0300369.

References

1. Baroni, M., Bernardini, S. BootCaT: Bootstrapping Corpora and Terms from the Web. In Proceedings of LREC 2004, pp. 1313-1316 (2004)
2. Benko, V.: Aranea: Yet Another Family of (Comparable) Web Corpora. In Petr Sojka, Aleš Horák, Ivan Kopeček and Karel Pala (Eds.): Text, Speech and Dialogue. 17th International Conference, TSD 2014, Brno, Czech Republic, September 8-12, 2014. Proceedings. LNCS 8655. Springer International Publishing Switzerland (2014)
3. Jongejan, B, and Dalianis, H.: Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP. Suntec, Singapore : Association for Computational Linguistics, pp. 145-153 (2009)
4. Kotiurova, I., and Trenina, P.: Comparative Analysis of Automatic POS Taggers Applied to German Learner Texts. 31st Conference of Open Innovations Association (FRUCT), 2022, pp. 115-124, <https://doi.org/10.23919/FRUCT54823.2022.9770886> (2022)
5. McDonald, R., Nivre, J., Quirmbach-Brundage, Y., Goldberg, Z., Das, D., Ganchev, K., Hall, K., Petrov, S., Zhang, H., Täckström, O., Bedini, C., Bertomeu Castelló, N., Lee, J. Universal Dependency Annotation for Multilingual Parsing. In Proceedings of ACL (2013)
6. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text Tokenisation Using unitok. In 8th Workshop on Recent Advances in Slavonic Natural Language Processing, Brno, Tribun EU, pp. 71-75 (2014)
7. Pomikálek, Jan. Removing boilerplate and duplicate content from web corpora. PhD thesis, Masaryk university, Faculty of informatics, Brno, Czech republic (2011)
8. Rasooli, M. S., Safari, P., Moloodi, A., Nourian, A.: The Persian Dependency Treebank Made Universal. 2020 (to appear)
9. Rychlý, P.: Manatee/Bonito – A Modular Corpus Manager. In: 1st Workshop on Recent Advances in Slavonic Natural Language Processing. Brno : Masaryk University, pp. 65-70. (2007)
10. Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. Proceedings of International Conference on New Methods in Language Processing, Manchester (1994)
11. Seraji, M., Ginter, F., Nivre, J.: Universal Dependencies for Persian. In Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC), pp. 2361-2365 (2016)

12. Straka, M., Hajič J., Straková J.: UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC), Portorož, Slovenia (2016)
13. Straková, J. Personal communication (2022)
14. Suchomel, V., Pomikálek, J. Efficient Web Crawling for Large Text Corpora. In Adam Kilgarriff, Serge Sharoff. Proceedings of the seventh Web as Corpus Workshop (WAC7). Lyon, pp. 39-43 (2012)
15. Vystrčilová, D., Khademi, M., Kříhová, Z., Novák, L. (2020, August 1–). Elektronická lexikální databáze indoíránských jazyků. Pilotní modul perština. Electronic Lexical Database of Indo-Iranian Languages. Pilot module: Persian. (Project No TL03000369, Technology Agency of the Czech Republic). Institute of Sociology, Czech Academy of Sciences & Faculty of Arts, Charles University. <https://www.soc.cas.cz/projekt/elektronicka-lexikalni-databaze-indoiranskych-jazyku-pilotni-modul-perstina>
16. Zadeh, B. Q., Rahimi, S.: Persian in MULTEXT-East Framework. In: Advances in Natural Language Processing, 5th International Conference on NLP, FinTAL 2006, Turku, Finland, August 23-25 (2006)

Pipeline Effectiveness in the Sketch Engine

Matúš Kostka

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00, Brno, Czech Republic
xkostka4@fi.muni.cz

Abstract. This paper focuses on measuring the effectiveness of the most used language pipelines (48 pipelines) in Sketch Engine for potential future efficiency improvements. This paper will describe the tool for parallel measuring made for this task, analyze the problem before measuring, analyze and represent results from measured data, and end with a conclusion.

Keywords: Pipelines, Tokens, Sketch Engine, Effectiveness, Language

1 Introduction

Sketch Engine supports approximately 100 languages which, of course, follows a similar number of pipelines [1]. Pipeline stands for a group of tools for text processing, like normalization, tokenization/segmentation, lemmatization and part-of-speech tagger, in one executable file. It is necessary for corpus creation. Every language has at least one pipeline. Because Sketch Engine uses several types of pipelines, most of them are by the Sketch Engine team. However, some came from different creators, meaning pipelines can differ in features, way of functioning, efficiency, CPU, and RAM consumption. Nevertheless, the amount of attention paid to the pipeline often depends on the size and usage of language, the pipeline was created for¹.

The effectiveness of pipelines is crucial for processing vast amounts of data and also for user satisfaction while creating their own corpora with Sketch Engine. So this paper focuses on the measurement of pipeline effectiveness in Sketch Engine to create an overview of the actual state of pipelines for potential future improvements. The measured parameters are **total execution time**, **CPU usage** and **memory usage**. The paper will briefly describe the problem of the measurements, the tool created for this task, selected pipelines, analyze of measured data and ends with a conclusion.

2 Closer description

The goal is to create a universal tool and statistics for better orientation in the efficiency of the pipelines in Sketch Engine. The measured parameters are

¹ For a full list of pipelines features, visit <https://www.sketchengine.eu/corpora-and-languages>.

execution time, **CPU usage** and **memory usage** (maximum resident set size). It was realized on files of a specified amount of tokens, in this case, **10,000**; **100,000** and **1,000,000** tokens for every file. These numbers were selected because Sketch Engine enables users to create their own corpora but with a default upper limit of 1 million tokens [1]. Measurement by the number of tokens and not by the file size was decided because each language has a different length of words. This means if the measurement were done by the file size, the number of tokens would not be the same for every language. The specified tokens files are made chiefly from data downloaded from Wikipedia for individual languages in 2020 and 2021 by a web crawler SpiderLing [2]. The measurement took place at one of the servers of Lexical Computing CZ, with 32 cores and 256 GB of ram. The result can be influenced by the background processes, the pipeline version, and the content of the used files on the server.

2.1 The tool

The script for this task is written in bash and can create a file of a given amount of tokens, measure execution time, CPU and RAM usage, and export measured data in CVS format for future processing. The script is based on UNIX command `/usr/bin/time`, which measures already mentioned parameters [8]. Creating a file with the requested amount of tokens is quite a time-consuming process because the data are first decompressed from gunzip format, tokenized, then the tokens are counted for a specified amount and turned back via **vert2plain** function to prevertical form. The script can work in 2 modes: creating a temporary file or with an already created file, to save time while repeating the measurement more time. It is recommended to run the script via makefile with `-j`, which will run several jobs simultaneously. The only limitation here is only the number of CPUs and cores the machine offers.

2.2 Used pipelines

Totally 48 pipelines are measured. Some languages are measured more times like Italian, French, Greek because they use more versions of pipelines, but on the other side traditional and simplified Chinese uses the same pipeline. These 48 pipelines are the most used pipelines in Sketch Engine and that is the reason why they were selected. There are several aspects causing that these pipelines can differ in results like the number of supporting features, way of implementation, type of alphabet, and unique language characteristics. When the pipeline support all features (mostly tagging and lemming) huge language models are loaded in the initialization phase, which can be time, CPU and RAM consuming. Model loading is crucial in pipelines for languages with a unique alphabet, like Chinese, Japanese, Arabian, Bulgarian and for languages similar to these. Bear in mind that quicker pipelines in the result can support fewer. In table 1, See Table 1, can be seen pipeline features with quick description.

Notes to Table 1, See Table 1:

Table 1: Pipeline composition

Feature name	Description
Uninorm	Normalization of text convert the content into NFKC normalization form. [5]
Unitok	Tokenisation of a text is the process of splitting the text into units suitable for further computational processing. It is an important data preparation step allowing to perform more advanced tasks. [4]
Lemmatizer	Lemmatization is a process of assigning a lemma to each word form in a corpus. [6]
Treetagger	Assigning special labels to each token in the corpus to indicate part of speech, grammatical categories. [7]

Note that features can differ for each pipeline. Like tokenization for Chinese, Japanese is called segmentation.

3 Analyze

It is evident that the amount of resources used is directly proportional to the number of tokens. Closer result from measurement with a million tokens can be seen in [1, 2, 3].

Table 2: Stats 10,000 tokens

	Min value	Max value	Average	Median
Execution time (sec)	2.47	762.7	78.27	54.46
CPU usage (%)	0	100	26	18
RAM usage (GB)	0.007	2.326	0.252	0.141

Notes to Table 2, See Table 2:

In the row of execution time, the minimum value was reached by **Hebrew pipeline** and the maximum value by **Tagalog pipeline**. In the CPU usage row, the minimum was also reached by **Hebrew pipeline** but the maximum by **Japanese pipeline**. And from the RAM point of view, the minimum of it was used by **Thai pipeline** and the maximum again by **Tagalog pipeline**.

Notes to Table 3, See Table 3:

The fastest execution time had **universal pipeline**, a default pipeline for languages without their pipeline. It supports just normalization and tokenization. The slowest was again **Tagalog pipeline**. From the CPU point of view, the least CPU resources were used by **Hebrew pipeline** and the most by **Japanese pipeline**. More than 100% of CPU usage is possible because the program is

Table 3: Stats 100,000 tokens

	Min value	Max value	Average	Median
Execution time (sec)	4.21	3300.14	271.02	108.76
CPU usage (%)	0	127	40	38
RAM usage (GB)	0.008	5.443	0.388	0.187

run on more cores (multiprocessing), 1 core == 100%. The maximum possible is 3200% because the server on which the measurement was realized have 32 cores. For more information, See <https://en.wikipedia.org/wiki/Multiprocessing>. And from the RAM point of view, the minimum of it was used by **Thai pipeline** and the maximum again by **Tagalog pipeline**.

Table 4: Stats 1,000,000 tokens

	Min value	Max value	Average	Median
Execution time (sec)	23.7	8109.08	1020.49	395.78
CPU usage (%)	0	222	75	77
RAM usage (GB)	0.008	5.629	0.733	0.209

Notes to Table 4, See Table 4:

The results are again similar as the previous two measurements and the fastest execution time was reached by **Universal pipeline** and the slowest by **Hebrew pipeline** (different pipeline as previous). The minimum of CPU resources was used by **Thai pipeline** and the most by **Italian pipeline**. And the RAM usage, minimum of it, was used by **Thai pipeline** and the most by **Japanese pipeline**.

4 Conclusions

The Tagalog pipeline has been the worst from the analysis of all three types of measurements. Even at the 1 million tokens measurement, it did not finish at all, the pipeline was repeatedly restarting itself. Also, the low usage of RAM and CPU by **Thai pipeline** (thai_sw1) and **Hebrew pipeline** (yap_he_v1) is questionable and the most probably not alright, but all repetitions of the measurements show similar values.

It is clear that pipelines for languages with different alphabets as Latin are usually slower. Also, the fact that 46% of pipelines are slower than 10 minutes is alarming and requires some attention [1]. The RAM usage is all right, only two pipelines use more than 5 GB, and those are tagalog_sw1 and mecab_unidic_comainu_jpn [2]. Moreover, from the CPU point of view, its evident that only 12% of pipelines are multithreaded [3].

One Positive fact is that all pipelines are in a linear relationship with the number of tokens. Hence, it is possible to calculate linear regression for quicker

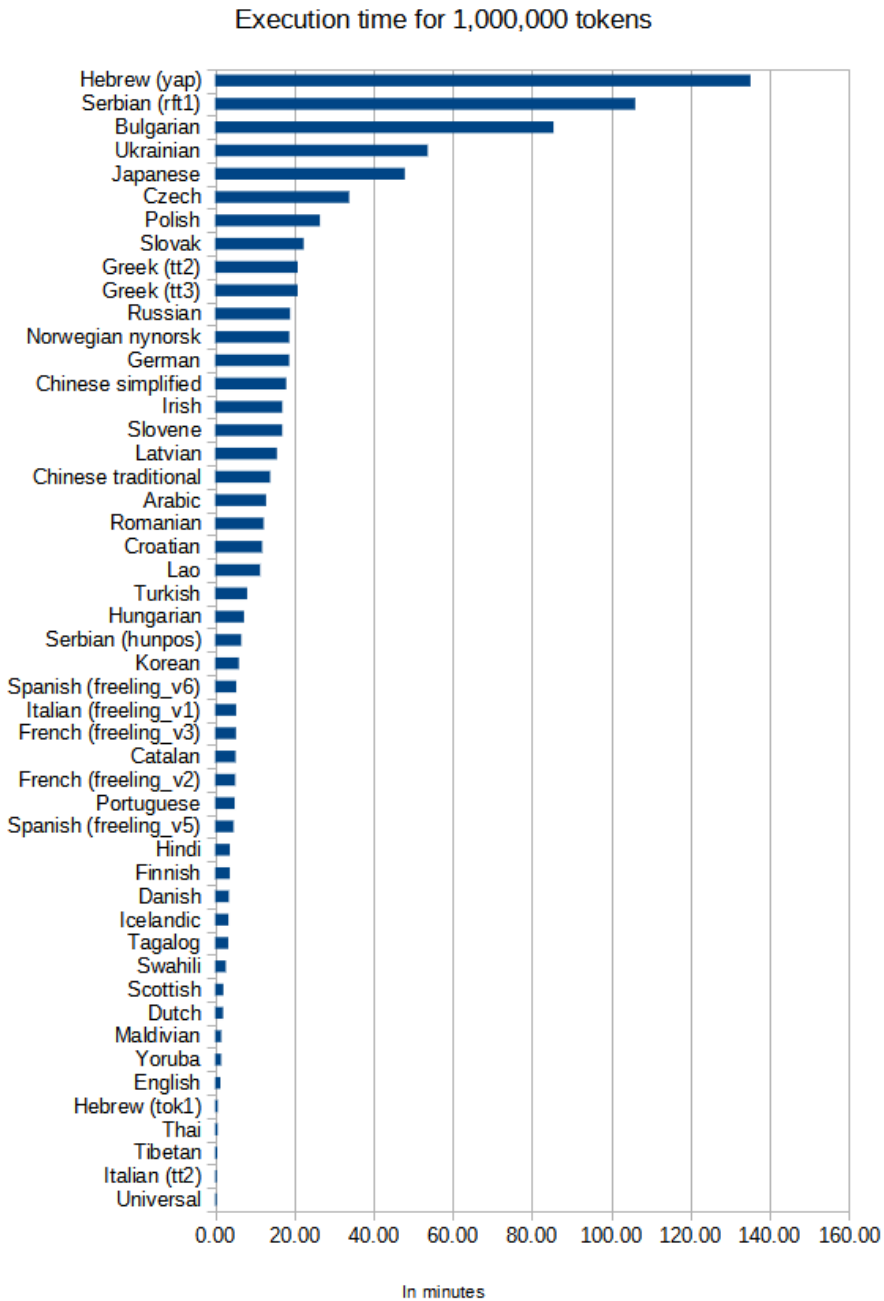


Fig. 1: Execution time for measured pipelines

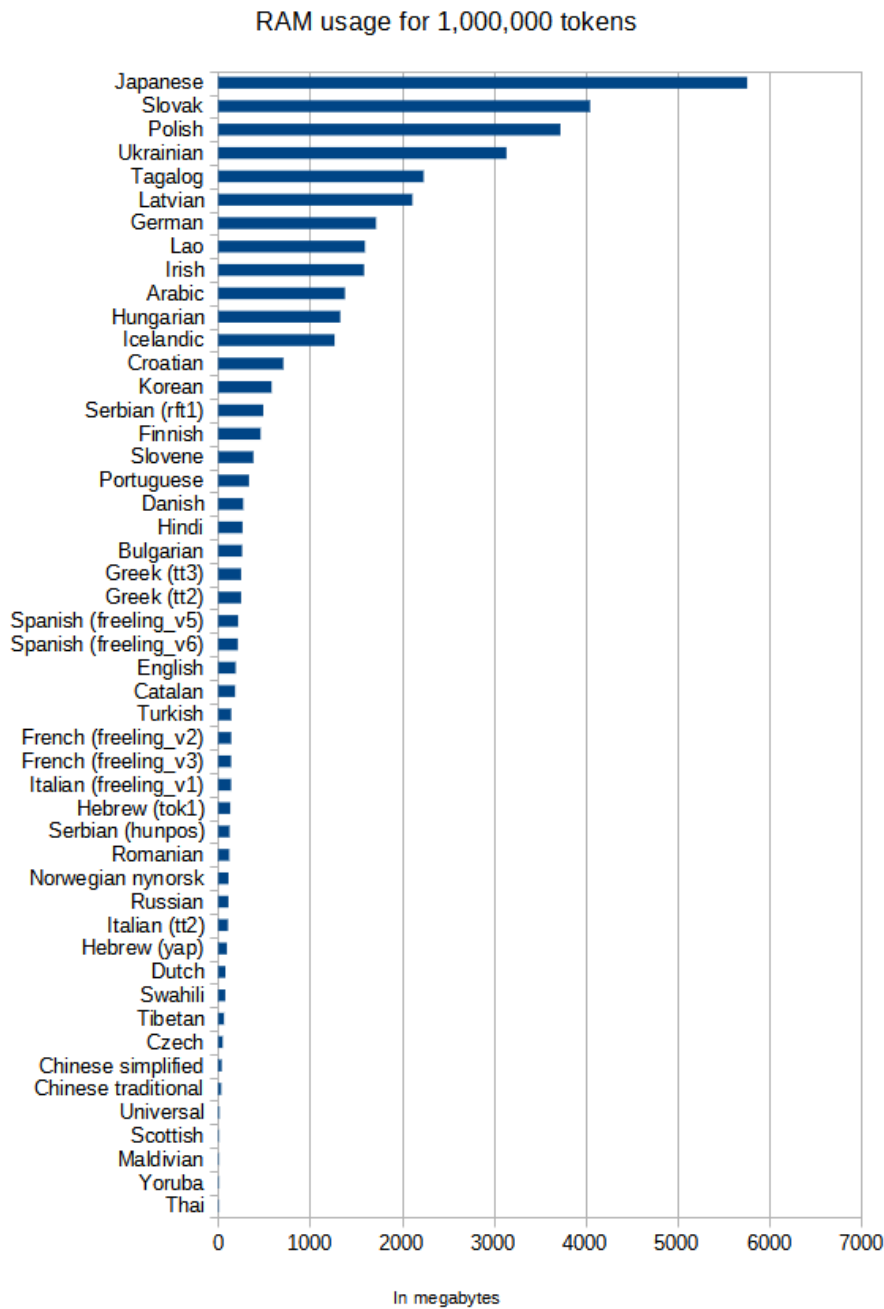


Fig. 2: RAM usage for measured pipelines

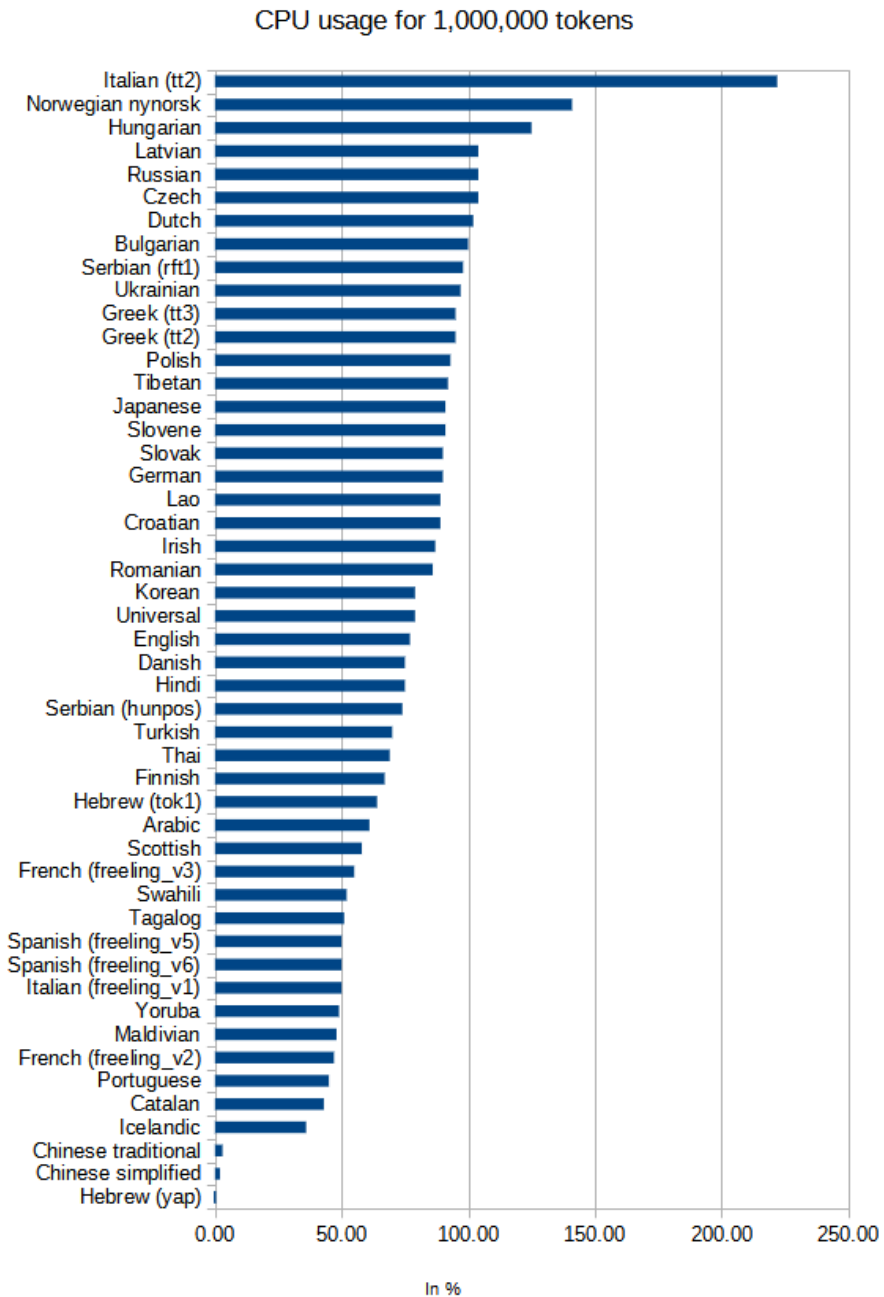


Fig. 3: CPU usage for measured pipelines

execution time estimation for a given token amount. However, it requires more measurements with a different number of tokens for a more precise result.

5 Possible future improvements and goals

The good idea is to realize more measurements with different amounts of tokens to have enough data to count linear regression with an acceptable result. Because now, there are just three measurements per pipeline from which the linear regression can be calculated but the error rate is quite high. The goal is to have at least ten measurements of different token numbers per pipeline. The next goal could be dividing pipelines according to supported features and measuring pipelines with similar features together because now it is highly probable that the fastest pipeline supports fewer features than the slower ones. And as the last goal and probably the most useful would be to measure the actual pipeline for all languages accessible in Sketch Engine.

Acknowledgments This publication has been fully supported by the Lexical Computing CZ s.r.o.

References

1. Kilgarrieff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., Suchomel, V.: The sketch Engine: ten years on. *Lexicography* 1(1), 17-19; 26-29 (2014)
2. Suchomel, V., Pomikálek, J.: Efficient web crawling for large text corpora. In: *Proceedings of the seventh Web as Corpus Workshop (WAC7)*. (2012) 39-43
3. Languages in Sketch Engine, <https://www.sketchengine.eu/corpora-and-languages/>. Last accessed 16 Nov 2022
4. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text Tokenisation Using unitok. (2014)
5. Unicode Normalization Forms, <https://unicode.org/reports/tr15/>. Last accessed 20 Nov 2022
6. Lemmatization, https://www.sketchengine.eu/my_keywords/lemmatization/. Last accessed 20 Nov 2022
7. POS tags, <https://www.sketchengine.eu/blog/pos-tags/>. Last accessed 20 Nov 2022
8. Time(1) — Linux manual page, <https://man7.org/linux/man-pages/man1/time.1.html>. Last accessed 20 Nov 2022

Constructing Datasets from Dialogue Data

Ondřej Sotolář , Jaromír Plhák , Michaela Lebedíková , Michał Tkaczyk , and David Šmahel 

Faculty of Informatics, Masaryk University,
Botanická 68a, 602 00, Brno, Czech Republic
{xsotolar, xplhak, x450458, x245062, davs}@fi.muni.cz

Abstract. We present methods for transforming raw dialogue data into a dataset suitable for processing with statistical NLP models. We reveal the potential pitfalls for processing this type of data, such as ensuring the representatives of the sample, the generalization ability of models, and the definition of the local context of the utterances. We use novel methods to solve these problems and demonstrate their effectiveness on an utterance classification problem. As a result, this paper provides guidelines for generating valuable datasets from dialogue data.

Keywords: Dialogue Dataset, Dataset Split, Online Conversations

1 Introduction

Online communication allows for the synchronous exchange of text, images, voice, and videos between two or more people [6]. It is realized using native applications such as Messenger, WhatsApp, and Discord, through social networks such as Facebook or Twitter, dedicated internet forums, and online discussions in general. Such communication is often studied in dialogue systems, which is concerned with designing agents (commonly called chatbots) that are capable of participating in the discourse [7]. In the case of dialogue systems, we have information available about the dialogue, such as the intents, topics, and dialog flow, because the agent actively shapes the discourse. In this work, we consider the general case, where we do not have this type of special information, and we work only with the text content of the dialogues and their metadata.

Our motivation for examining a sample of online dialogues is the discovery of phenomena of interest. Doing so with conventional and manual methods is inefficient for two reasons: a sample of a significant size includes a large quantity of unstructured text, and the occurrence of the researched phenomena might be low [14]. While this goal is similar to intent detection, as known in dialogue systems, using the existing methods is usually impossible. In the general case, we lack information on the structure of the dialogues, which the conversational agent controls. Information on how to approach Natural Language Processing (NLP) tasks in the general case is scattered among small, often unrelated tasks (such as Dialogue Act Recognition [11,10], Argument Mining [12,5,2,9], Short

Text Classification [8,4], or Emotion Detection [1]). This makes it difficult to research methodologies for novel tasks.

This paper aims to present a practical methodology for processing raw dialogue data. We provide guidelines with examples, diagrams, and algorithms for the complete process of generating datasets from dialogue data. In Section 2, we present a practical methodology for storing and retrieving the dialogues. Section 3 presents an algorithm for constructing training examples with a local context. Finally, in Section 4, we propose a novel algorithm for k -fold data splitting.

2 Data Structure for Storing and Retrieving Dialogues

Dialogues are composed of temporal sequences of *utterances* as shown in Table 1. Utterances are typically short text fragments, complete sentences, or short sequences thereof. They are uniquely identified by their timestamp within the dialogue. Metadata associated with individual utterances can be, for example, the author, a reference to previous utterances (identifying a response), and others. Many applications allow to use multi-media in the dialogues; however, we omit them here.

Furthermore, in this work, we consider the data to be annotated with class labels at the utterance level. The labels identify either syntactic or semantic phenomena per their definition. Whether the dialogue context influences the utterance labels is a design decision. In the following, we encode the *none* class with index 0 and others with $\{1..N\}$.

Table 1: Excerpt from a dialogue from the dataset from [13] (translated to English) showing time, participant name, the utterance, and the phenomenon label in each row.

Time	Author	Utterance	Class
01:44:07	John	I'll finish the Math task tomorrow	none
01:44:13	John	Like, I really have to do it	none
01:44:28	Tim	The math task looks easy to me	Emotional Support
01:44:49	Tim	You have 6 hours to deadline, chill	Emotional Support
01:46:34	John	But I'm really tired after the day	none
01:47:51	Tim	I'm having some tea and I'm super	none

To efficiently store and retrieve dialogues, we propose a hierarchical data structure that reflects the relationships between the dialog components. At the top level, we can, in many cases, identify an owner, first author, or originator of the dialogue. In Instant Messaging (IM), it is a user; in forums and social media, this would be the original poster (the topic creator). The second level of the structure is composed of *threads* that group dialogues together. The set of utterance authors in a thread is unique. Because threads might be long-running, we suggest a third level of the structure. There, we delimit individual

conversations with a time constraint to help separate different topics. We argue that in long-running threads, after a certain pause, the topic is more likely to change. We suggest finding the threshold for separating the conversations experimentally. For example, with IM conversations, we have used 1-hour long pauses to delimit them. The resulting structure is shown in Figure 1.

For practical reasons, we map this structure to a table in a relational database shown in Figure 2. In Figure 3, we show an example PostgreSQL query for retrieving dialogue data. It is structured into conversations, where each row of the result contains one conversation with its utterances and labels in ordered lists. Any other metadata can be retrieved similarly.

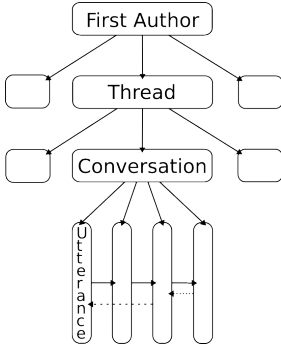


Fig. 1: Data structure for dialogues segmented by the first author, threads, time-delimited conversations, and utterances.

column_name	type
<u>uid_first_author</u>	integer
<u>thread_id</u>	integer
<u>conversation_id</u>	integer
<u>line_num</u>	integer
<u>time</u>	timestamp
<u>uid_author</u>	integer
<u>reaction_to</u>	integer
<u>label</u>	integer
<u>text</u>	varchar

Fig. 2: Relational database table for structure shown in Figure 1.

```

SELECT
  ARRAY_AGG(d.line_text::TEXT ORDER BY line_num asc) AS texts,
  ARRAY_AGG(d.label::INTEGER ORDER BY line_num asc) AS labels
FROM dialogues s
WHERE d.label = 0 OR d.label = 1
GROUP BY d.uid_first_author, d.thread_id, d.conversation_id;

```

Fig. 3: Example of a SQL query (PostgreSQL) to retrieve the time-delimited conversations.

3 Constructing Training Examples with Local Context

To construct training examples, we could use individual utterances. However, previous research [3,10,4,9] has shown that including the context of the dialogue can improve the solutions for many different tasks. In this work, where we consider utterance-level labels, we also use the concept of *local context*. The local context of a target utterance is defined as a window of the neighboring utterances. The size and position of the context window is a design decision. It is called local context because the window size usually covers only a few utterances, and its purpose is to help the model to capture local dependencies. This contrasts with other types of context, such as the whole dialogue or the language style of a particular user across many dialogues. Such types of context often span a large amount of text, which has to be condensed due to the practical limits of sequential models used in NLP. Conversely, local context can usually be used in its original text form.

We present an algorithm for delimiting the local context of a target utterance in Algorithm 1. We use the sliding window concept. We found that individual utterances differ significantly in their character length; thus, we do not define the threshold with an utterance count but with a character count instead. In our algorithm, the context window includes an arbitrary number of neighboring utterances as long as the sum of their character lengths does not exceed the given limit. The limit is soft: if the character limit is reached within the bounds of an utterance, it is appended as a whole, thus exceeding the limit. The algorithm constitutes a complete solution to generate the training dataset if iterated over selected conversations (or whole threads).

Algorithm 1: Algorithm for constructing training examples.

function rows_to_examples

Input : *rows*: list of utterance tuples (text, label $\in \{0..N\}$, metadata) ordered by time (conversation or whole thread),
ctx: character length of context,
sep: utterance separator,
sep_{target}: separator of target utterance,
has_small_pieces: stop if window reaches start & contains all 0s.

Output: *examples*: set of examples as tuples (text, label, metadata)

examples $\leftarrow \emptyset$;

for *t, l* in *rows* **do**

if *len(t)* < 2 **then** # Case when only one utterance in dialogue
 # Compose function concatenates the utterance data in the window (uses separators for text).
 examples.add(compose_example(*t, l, sep, sep_{target}*));
 continue

target_i $\leftarrow \text{len}(t) - 1$;

while *target_i* > 0 **do** # Case when 2 and more utterances

last_i $\leftarrow \text{target}_i$;

char_{cnt} $\leftarrow 0$;

while *last_i* > 0 & *char_{cnt}* < *ctx* **do**

last_i $\leftarrow last_i - 1$;

char_{cnt} $\leftarrow char_{cnt} + \text{len}(t[last_i])$;

examples.add(
 compose_example(*t[last_i : target_i + 1], l[last_i : target_i + 1], sep, sep_{target}*));

if *has_small_pieces* & *last_i* = 0 & *sum(l[0 : target_i])* = 0 **then**
 break;

target_i $\leftarrow target_i - 1$;

return *examples*;

4 Selecting Representative Samples

The performance of predictive models is measured with a testing sample that is unseen during training. The measurement reliability depends on the validity of this sample, which should be representative of the actual data, to determine whether the model overfits the training sample. With dialogues, this concerns the style of individual authors and also the topics of the conversations, which might have a distinct vocabulary. We argue that splitting such data naively into the training and testing samples may positively bias the performance measurement, thus not reflecting the model’s true generalization ability. Imagine a user who authors many utterances on a single topic with distinct keywords. Consider a classification task: if we split this data between the train and test samples, we risk overfitting the model on the keywords, then also successfully classifying the examples in the test set. This would result in a model with a good measured performance but a poor generalization ability because it would likely fail on data from other users.

To avoid this issue, we first suggest analyzing the data with regard to the contributions of individual utterance authors. In Figure 4, we show different samples of a dataset from [13], each annotated with a different class. We would assume that sample a) *is not* representative and sample b) *is* representative of real-world data based on the distribution of the contributions of different authors of the utterances.

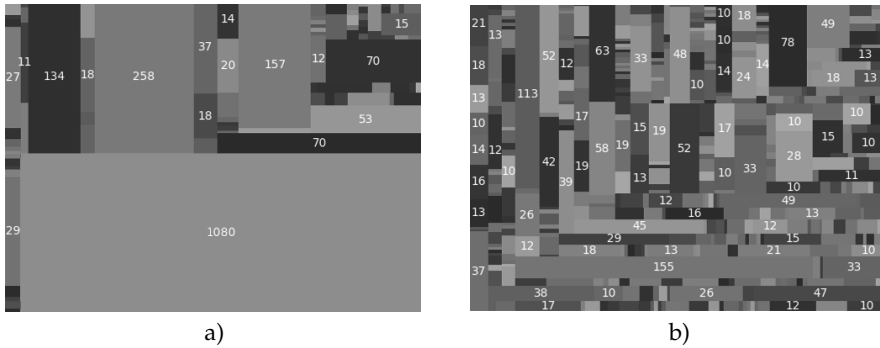


Fig. 4: The content contributions in samples of the IM dataset from [13], each labeled with a particular class. One rectangle represents a person. The relative size of the rectangle and the number within each rectangle represent the number of utterances they authored.

Second, we suggest using k -fold cross-validation. To use it, we need to split the data in a stratified manner for a given k . However, standard splitting algorithms are unsuitable for dialogues because we need specific criteria to define the splits. Ideally, we would have examples with disjoint sets of authors in each split. This is not always necessary; thus, we need a parameter for setting the maximal overlap of the example’s authorship between the splits (max_{c_rank}).

Furthermore, following the idea of stratified sampling, the splitting algorithm should keep the same size of the splits and also the same class ratio.

In Algorithms 2, 3, and 4, we present the *dialog_k_fold* algorithm, which splits the data into k different groups per the specified criteria. If such a split exists, the algorithm will return the number of splits given by the k parameter. Otherwise, it will return the maximum possible splits for the criteria plus the set of remaining examples. We formalize the condition for proving its effectiveness in Lemma 1.

Algorithm 2: Algorithm for modified k -fold split for dialogue data.

function *dialog_k_fold*

Input : E : set of examples,

k : desired number of splits,

max_{c_rank} : threshold for maximum number of author overlap

Output : *splits*: list of 1.. k data splits

remainder: examples excluded from the split groups

$T \leftarrow \text{group_by_author_tuples}(E)$;

$G \leftarrow (g_1, \dots, g_k)$;

$R \leftarrow \emptyset$;

while $\exists t \in T : \neg \exists g \in G : t \in g$ **do**

$t \leftarrow t$ with $\min(t.c_rank)$, **if tied then** use $\max(t.size)$;

$g \leftarrow \text{best_group}(G, t, max_{c_rank}, \frac{size(E)}{k}, label_ratio(E))$;

$g.add(t)$;

$R \leftarrow T \setminus \{t : t \in g, g \in G\}$;

return G, R ;

Lemma 1. Given a dataset \mathcal{D} , a model performance measurement $M(\text{train}, \text{test})$, let $(d_1, d_2, d_3) \leftarrow \text{dialog_k_fold}(\mathcal{D}, k = 3)$ create three splits, where the overlap between utterance authors is minimal. Set $\mathcal{D}_{\text{holdout}} \leftarrow d_3$ aside as a holdout set and merge the rest $\mathcal{D}_{\text{new}} \leftarrow d_1 \cup d_2$. Let:

$$\mathcal{S}_{\text{rand}} = (r_1, r_2) \leftarrow \text{random_split}(\mathcal{D}_{\text{new}}, k = 2), \quad (1)$$

$$\mathcal{S}_{\text{dkf}} = (f_1, f_2) \leftarrow \text{dialog_k_fold}(\mathcal{D}_{\text{new}}, k = 2). \quad (2)$$

Then, the (cross-validated) difference between $M(r_x, r_y)$ and $M(r_x, \mathcal{D}_{\text{holdout}})$ should be greater than if we use \mathcal{S}_{dkf} for training. The following condition should hold up to additional cross-validation, i.e. for all permutations of (d_1, d_2, d_3) :

$$\text{avg}|M(r_x, r_y) - M(r_x, \mathcal{D}_{\text{holdout}})| > \text{avg}|M(f_x, f_y) - M(f_x, \mathcal{D}_{\text{holdout}})|, \quad (3)$$

where:

$$x, y \in \{1, 2\}, x \neq y.$$

Proof. We prove Lemma 1 experimentally using the dataset sample b) presented in Figure 4, which shows the utterance author distribution. The results presented in Figure 5 demonstrate that condition (3) of Lemma 1 holds.

Algorithm 3: Algorithm for grouping examples by author tuples. Additionally, it computes author overlap and ranks each group by its severity.

```

function group_by_author_tuples
  Input :  $Examples(thead\_id, labels, lines\_per\_author)$ : set of examples
  Output :  $T$ : set of grouped examples with aggregate attributes
   $T \leftarrow$  group  $examples$  by author tuples with aggregates:
     $size \leftarrow count(labels)$ ,
     $ratio \leftarrow$  ratio of labels in the group,
     $authors \leftarrow$  indexed list with sum of each author's utterance count.
  for  $t \in T$  do # Calculate utterance authors overlap into an n-hot
    vector
    for  $a \in t.authors$  do
      for  $g_i \in T, g_i \neq t$  do
         $t.conflicts[i] +=$  if  $a \in g_i.authors$  then  $t.count(a)$  else 0;
  # Finally, we rank the groups in  $T$ . The rank values can have
  duplicates.
  foreach  $t \in T$ :  $t.c\_rank \leftarrow$  order of  $t$  in  $T$  ordered by  $sum(t.conflicts)$ ;
  return  $T$ ;

```

Algorithm 4: Algorithm for selecting the split group to add the given author group.

```

function best_group
  Input :  $G$ : split groups,
     $t$ : group of examples (grouped by author tuple),
     $max_{c\_rank}$ : threshold for maximal author overlap,
     $size_{desired}$ : desired group size: ideally  $\frac{size(dataset)}{k}$ ,
     $class\_ratio_{desired}$ : desired class ratio: ideally same as original dataset
  Output :  $g$ : chosen split group
   $g_m \leftarrow$  group with minimum conflicts with  $t$ ;
  if  $g_m.conflicts(t) > max_{c\_rank}$  then
    return  $\emptyset$ ;
   $g_n \leftarrow$  group with maximum  $|g_n.size - size_{desired}|$  with  $t$ ;
   $g_o \leftarrow$  group with maximum  $|g_n.class\_ratio - class\_ratio_{desired}|$  with  $t$ ;
   $g \leftarrow$  select from  $\{g_m, g_n, g_o\}$  with most votes, if tied then take  $g_m$ ;
  return  $g$ 

```

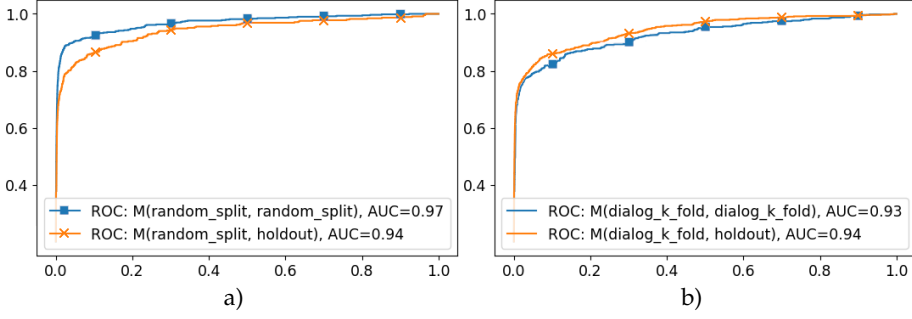


Fig. 5: Model performance comparison using a performance metric \mathcal{M} (see Lemma 1) and a holdout set $\mathcal{D}_{holdout}$, where utterances are authored by a disjoint set of authors than in \mathcal{D}_{new} . In a), \mathcal{D}_{new} is randomly split into (r_x, r_y) . The model trained and tested on (r_x, r_y) shows artificially higher measured performance than the more reliable measured performance of the same model tested on $\mathcal{D}_{holdout}$. In b), where \mathcal{D}_{new} is split using *dialog_k_fold*, we can see the significantly closer performance, proving the effectiveness of our algorithm.

5 Discussion and Limitations

We have experimentally proven that *dialog_k_fold* effectively improves the measured performance reliability. However, we have not given formal proof of the algorithm correctness or time complexity. We leave this for future work. Furthermore, the function *best_group* defined in Algorithm 4 leads to a multi-criteria optimization problem, which in our current implementation, we have solved with a rule-based, heuristic approach. We suggest finding an optimal solution in further work. The function also needs to calculate the authorship overlap of each utterance with each other in $O(n^2)$ time which is expensive for large datasets.

6 Conclusion

We have presented practical methods for structuring, storing, and retrieving dialogue data. We have also presented an algorithm for constructing training examples from such data. Furthermore, we presented a novel *k*-fold algorithm for the stratified splitting of datasets of dialogue data. We have demonstrated that using our *dialog_k_fold* algorithm improves the reliability of performance measurements when compared to naive splitting methods.

Acknowledgements. This work has received funding from the Czech Science Foundation, project no. 19-27828X.

References

1. Acheampong, F.A., Wenyu, C., Nunoo-Mensah, H.: Text-based emotion detection: Advances, challenges, and opportunities. *Engineering Reports* (7), e12189 (2020)
2. Aker, A., Sliwa, A., Ma, Y., Lui, R., Borad, N., Ziyaei, S., Ghobadi, M.: What works and what does not: Classifier and feature analysis for argument mining. In: *Proceedings of the 4th Workshop on Argument Mining*. pp. 91–96 (2017)
3. Chatterjee, A., Narahari, K.N., Joshi, M., Agrawal, P.: Semeval-2019 task 3: Emocontext contextual emotion detection in text. In: *Proceedings of the 13th international workshop on semantic evaluation*. pp. 39–48 (2019)
4. Chen, J., Hu, Y., Liu, J., Xiao, Y., Jiang, H.: Deep short text classification with knowledge powered attention. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. vol. 33, pp. 6252–6259 (2019)
5. Habernal, I., Gurevych, I.: Argumentation mining in user-generated web discourse. *Computational Linguistics* (1), 125–179 (2017)
6. Huang, H., Leung, L.: Instant messaging addiction among teenagers in china: Shyness, alienation, and academic performance decrement. *CyberPsychology & Behavior* (6), 675–679 (2009)
7. Jurafsky, D., Martin, J.H.: Chapter 24: Chatbots and dialogue systems in speech and language processing. vol. 3. US: Prentice Hall (2014)
8. Lee, J.Y., Dérmoncourt, F.: Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827* (2016)
9. Lugini, L., Litman, D.: Contextual argument component classification for class discussions. *arXiv e-prints* pp. arXiv–2102 (2021)
10. Martínek, J., Cerisara, C., Král, P., Lenc, L.: Cross-lingual approaches for task-specific dialogue act recognition. In: *IFIP International Conference on Artificial Intelligence Applications and Innovations*. pp. 232–242. Springer (2021)
11. Martínek, J., Král, P., Lenc, L., Cerisara, C.: Multi-lingual dialogue act recognition with deep learning methods. *arXiv preprint arXiv:1904.05606* (2019)
12. Persing, I., Ng, V.: End-to-end argumentation mining in student essays. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pp. 1384–1394 (2016)
13. Sotolář, O., Plhák, J., Šmahel, D.: Towards personal data anonymization for social messaging. In: *International Conference on Text, Speech, and Dialogue*. pp. 281–292. Springer (2021)
14. Sotolář, O., Plhák, J., Tkaczyk, M., Lebedíková, M., Šmahel, D.: Detecting online risks and supportive interaction in instant messenger conversations using czech transformers. *RASLAN 2021 Recent Advances in Slavonic Natural Language Processing* p. 19 (2021)

Semi-Manual Annotation of Topics and Genres in Web Corpora The Cheap and Fast Way

Vít Suchomel^{†‡}, Jan Kraus[‡]

[†]Natural Language Processing Centre
Faculty of Informatics, Masaryk University, Brno, Czechia

[‡]Lexical Computing
Brno, Czechia
`name.surname@sketchengine.eu`

Abstract. In this paper we present a cheap and fast semi-manual approach to annotation of topics and genres in web corpora.

The main feature of our method is assigning the same topic or genre label to all web pages coming from websites most represented in the corpus. We assume that web pages within a site share the topic of the whole domain. According to the evaluation of texts coming from sites that were manually assigned a topic label, our hypothesis holds in 92 % of cases. In other words, the noise in these semi-manually labelled web pages is just 8 %. That is clean enough to train a classifier of texts from websites not seen in the process.

The procedure of fast manual topic and genre labelling of web domains is described in this paper. Recommendations for training a topic or genre classifier using semi-manually labelled texts from large websites follow.

Keywords: web corpus, text corpus, topic, genre, text annotation

1 Introduction and Motivation

According to [1], text corpora built from texts from the internet are used for language modelling, information retrieval, question answering, automatic population of ontologies, translating terms and language teaching. Text collections large enough to find evidence of scarce language phenomena in natural language context have to be compiled from the largest, free and easy-to-use data source – the web.

Understanding the sources of a web corpus and its content is important for users of web corpora. However, the internet is not organized by linguistic properties of the text or text types so one has to add the desired metadata by manual or automated ways. This paper is based on our experience with adding information about text topic and genre to web pages in the TenTen family of web corpora [2] for corpus manager Sketch Engine [3].

Once the information about text types is inserted, Sketch Engine allows the users to see the size of data within a corpus, as document count, sentence count, or token count by topic or by genre. For example, genre *news* or topic *religion* usually belong to the most represented text types in corpora in languages with a small presence on the web.

The users can also limit their search queries or other corpus based analyses to a subcorpus made from a single text type or from a combination of text types. For example, terminologists may require working just with documents labelled by topic *nature & environment*. Another example could be a language model for a typing prediction software. The model would benefit from texts labelled by genre *discussion*. On the other hand, genre *legal* should be avoided or reduced in this case of corpus use.

This paper is a follow-up to our recent work on semi-manual methods of computer generated text removal and annotation of topics and genres in web corpora. We proposed a semi-manual approach consisting of manually checking the largest sources of data and training a non-text classifier, using this data, for the rest of the corpus in [4, p. 85]. Our assumption in [5] was that all pages in a web domain shared the same properties with regards to text quality. We noted such hypothesis could lead to mistakes and noisy training data for a text quality classifier while there were two clear advantages of the approach: Millions of training samples for the classifier and a low cost of manually annotating the whole websites. [6] applied the approach to document metadata, namely the text topic.

An extension of the method to both topics and genres is described in this paper. Our aim is to reliably annotate a large part of a web corpus with only a small human effort, thus cheaply. The procedure of fast manual topic and genre labelling of web domains is documented in detail in the following chapters.

2 Determining a Set of Topics and Genres Feasible to Recognize

We understand the topic of a text as its subject, recognizable mostly by lexical properties of the text, i.e. its words. The genre is determined by both syntactic and lexical features of the text, i.e. defined by the style of writing.

While Dewey Decimal Classification provides a wide tree of subjects or text topics (by design ten main categories, each with ten subcategories and each with ten third level labels) and while there are 24 main genres with 31 subgenres in BNC 1994 or 8 main genres with 37 subgenres in BNC 2014 [7], web corpora are not constructed in a deliberate way and the internet is not populated by texts selected in order to belong to a pre-designed topic or genre hierarchy.

When determining topic or genre of web texts, we have to deal with large grey zones between class definitions. There are texts belonging to multiple classes, e.g. a post about a recent release of a football computer game in a personal site. – Is it a news, a blog, or both? Is the topic sports, games, or

both? How much text can form a separate topic to recognize in a multi-topic document?

To address the issue of grey zones and find a set of topics and genres feasible to recognize, we merged the definition of categories with the process of manually labelling texts. We started with a large English web corpus¹, the list of topics in web directory Curlie.org (formerly DMOZ.org)² and with Sharoff's Functional text dimensions for large web corpora [8]. We merged categories that caused problems to decide in which of them real web documents belong or where there was a small number of such texts, even though their definition in an annotation manual seemed clear. The real texts were just neither white nor black but grey. We did not want to keep labels with a low annotator agreement.

On the other hand, we introduced topic labels that were easy to recognize. We added more words in category names too to help understand which content belongs there. This approach is comparable to text types in Estonian National Corpus – [9, p. 215–216], in fact we were inspired by some of their topics but since we wanted to keep a high content variety within each class, labels assigned to documents from a low number of websites were discarded – that is why we kept less categories than ENC in the end.

The list of topics and genres recognized in the English web corpus can be found in Table 1 and in Table 2, respectively. Classes with a low number of source websites were disregarded. When we applied the same label selection process to smaller corpora (in other languages than English), more classes were discarded to keep a variety of sources within each class.

3 Fast Manual Topic and Genre Labelling of Web Domains

The procedure of manual topic and genre labelling of the content of whole websites follows.

First, web domains represented in the corpus are ranked by the count of tokens they contributed. Top ranking sites, i.e. the largest sources of text, are examined thoroughly while the time spent by examining smaller sources drops with the domain size. 3,000 largest domains in the English web corpus were inspected. These sources cover 40 % of corpus tokens. For other languages, the count depends on the corpus size – usually between 300 and 1,500 largest domains, covering at least 60 % and even up to 90 % of corpora – by checking just a small part of websites to keep the process efficient.

Documents from a single domain sharing frequent prefixes of paths can be examined separately, independently on the rest of documents within the domain, to adapt to websites with multiple topics. This technique works for sites with path prefixes such as `"/sports/"`, `"/culture/"`, etc.

¹ The corpus was enTenTen21, obtained from the web in 2021, comprising of 65 billion tokens at this stage of processing in which sources of bad texts are identified and text types are determined.

² <https://curlie.org/>

Table 1: Topics recongized in a large English web corpus. Out of top 3,000 websites that were inspected, 887 were assigned a topic. Note four categories marked by the red colour that were not represented by enough websites so their labels were discarded in the final revision of the data.

Topic	Websites	Tokens
arts	12	169 655 242
beauty & women	6	45 899 006
cars & bikes	49	268 201 168
construction & real estate	1	4 610 212
culture & entertainment	123	695 609 769
economy, finance & business	62	387 271 125
education	15	79 155 574
food & drinks	2	9 774 572
gambling & casinos	1	7 839 308
games	52	324 004 431
health	59	426 786 724
history	24	176 510 675
hobbies	18	111 828 110
home, family & children	7	47 126 547
lifestyle	0	0
nature & environment	6	64 495 602
pets & animals	9	33 432 198
politics & government	27	243 239 797
reference/encyclopedias	10	4 210 237 110
religion	71	424 919 420
science	51	594 461 579
sex	10	209 398 259
sports	103	647 268 352
technology & IT	138	887 566 212
travel & tourism	31	162 020 069
Total	887	10 231 311 061

Table 2: Genres recognized in a large English web corpus. Out of top 3,000 websites that were inspected, 611 were assigned a genre.

Genre	Websites	Tokens
blog	99	748 208 188
discussion	194	1 327 118 539
fiction	55	1 009 319 746
legal	37	507 984 084
news	226	1 284 058 175
Total	611	4 876 688 732

Second, an annotator records the topic and/or the genre in an inspection table. The table is generated by a script from the list of largest sources of the corpus provided by the corpus manager. Each row of the table is dedicated to inspecting one website. To increase the efficiency of the process, the quality of the site content is checked in this phase, together with determining text types.

There are the following columns in the table:

1. The hostname (e.g. "www.bbc.com") – Names with suspicious or long words, generic or foreign TLDs, language code in TLD are checked for generated content.
2. A link to the landing page of the site (e.g. "https://www.bbc.com/") – The page is checked in a web browser for low quality text, no text, hijacked/unrelated content, selectors with too many language mutations (high chance of machine translated (MT) content, MT scripts in the source code. A dead site is suspicious too (a high quality content does not get shut down often).
3. A link to 100 random triples of consecutive sentences in context displayed in Sketch Engine – 3 to 10 sentence triples are inspected, the rest is briefly seen and consulted more in case of doubtful content in the sentences that were read well or in case of a dubious hostname or a suspicious live site. Machine generated text, clusters of nonsense characters, unrelated phrases stitched together are indicators of bad content and lead to the removal of the whole source from the corpus. Each sentence triple can be tracked to the original web page within the domain (if the page still exists) to see the live content in a web browser.
4. Topic and genre – The annotator should be able to decide if the content shows lexical or syntactic features typical for a recognized text type. No label is given if the person is not sure. No class is given instead of assigning multiple labels to pages sites with many text types.
5. The size of the site in tokens – used to estimate how much time should be spent inspecting the source.

The procedure does not require an expert linguist or computer scientist. Any person with a bit of a sense of language and a common web browsing skill is sufficient. The inspection table is a guide easy to follow. In our experience, the

annotator does not even need to understand the language after some exposure to the task in a language they are familiar with. Photos in live pages tell much about the topic, e.g. sports or health, and the structure of the page can indicate the genre, e.g. a discussion forum, without understanding a word. Browser plugins connected to Google Translate or DeepL translating the page content help in other cases.

Depending on the rank of the website, a human annotator can spend between several minutes to as less as 20 seconds with each item to inspect. Not assigning any labels is encouraged to reduce noise in the annotations. Altogether, the procedure is quite efficient because it is fast and simple:

- A large part of a corpus is covered just by checking a small amount of random sentences from the most contributing sources and checking the live sites,
- all documents from a website are labelled with the same text type as the whole site,
- and no expert skill is required.

4 All for One, and One for All? The Evaluation

[6] compared manually assigned topic labels to 960 documents (a label assigned separately to each document, regardless the site labels) with the labels of their source sites. The document labels reportedly matched the respective domain labels in 92 % of cases. An estimated noise of just 8 % in the annotated data enabled training a topic classifier using the semi-manually obtained labels.

5 Conclusions

In this paper we presented a semi-manual procedure to annotate topics and genres in large web corpora. Unlike manually inspecting each sample and training a classifier on 100 % clean data – which is the usual approach – our method relies on seeing just random sentences and a few live web pages to represent up to tens of thousands of texts. The scheme was designed to decrease the time an annotator needs to check one website.

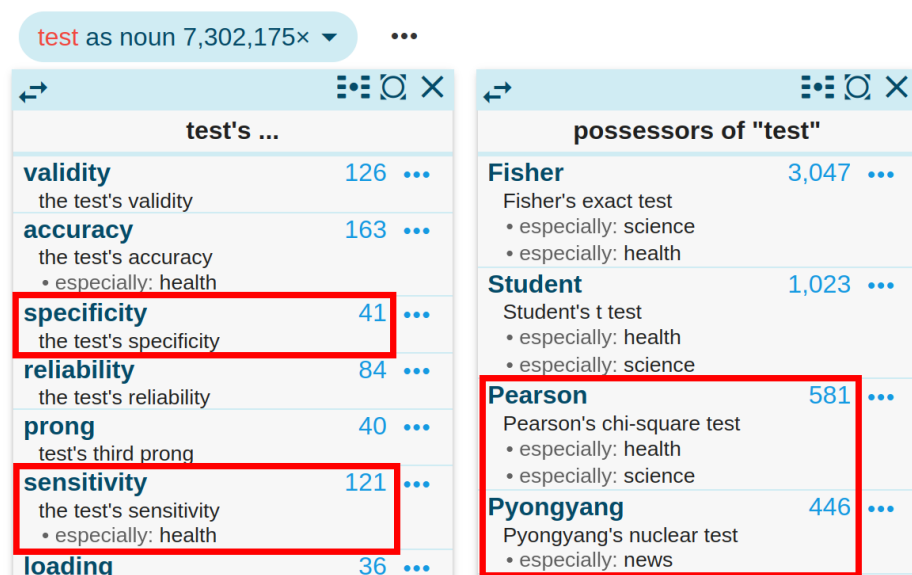
According to the evaluation, the noise in the labelled texts is small enough to allow using the data for training a text type classifier. The classifier can be applied to documents coming from websites that were not inspected manually thus covering the whole corpus. We recommend to balance the training samples in order to keep a wide diversity of the set, e.g. by limiting the count of documents from a single web domain. We also suggest disregarding classes consisting of samples from less than 10 websites. Alternatively, one can find additional sites in the ranked list just to boost the size and variety of under-represented text types.

An example of a language analysis benefiting from text type labels produced by our method can be seen in Figure 1. Word Sketch, the report shown in the screenshot, is used by publishing houses to produce dictionaries.

The main contribution of our work is showing how to reliably annotate a large part of a web corpus with only a small human effort.

Fig. 1: Word Sketch of noun *test* in a large English web corpus from 2020. Frequent phrases and frequent text types are shown. The counts of co-occurrences of “test” with collocates are displayed too. Note the phrase “the test’s sensitivity” is specific to topic health while the phrase “the test’s specificity” is not more specific to topic health than to topic science. Indeed, health researchers seem to be more interested in the sensitivity of tests than general researchers. Also note that “Pearson chi-square tests” occur usually in texts on health or science while “Pyongyang’s nuclear tests” can usually be found in the news. The information about text types in Word Sketches is appreciated by lexicographers.

WORD SKETCH



Acknowledgements This work has been partly supported by the Ministry of Education of CR within the Lindat Clarin Center. This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731015.

References

1. Kilgarrriff, A., Grefenstette, G.: Introduction to the special issue on the web as corpus. *Computational linguistics* 29(3) (2003) 333–347

2. Jakubíček, M., Kilgarriř, A., Kovář, V., Rychlý, P., Suchomel, V.: The TenTen Corpus Family. International Conference on Corpus Linguistics, Lancaster (2013)
3. Kilgarriř, A., Baisa, V., Buřta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., Suchomel, V.: The Sketch Engine: ten years on. *Lexicography* **1** (2014)
4. Suchomel, V.: Better Web Corpora For Corpus Linguistics And NLP. PhD thesis, Masaryk University (2020)
5. Suchomel, V., Kraus, J.: Website properties in relation to the quality of text extracted for web corpora. In: The Fifteenth Workshop on Recent Advances in Slavonic Natural Language Processing. (2021) 167–175
6. Papčo, R.: Topic classification for web corpora: Method comparison and crosslingual transfer. Master's thesis, Masaryk University (2022) Supervisor: V. Suchomel.
7. Brezina, V., Hawtin, A., McEnery, T.: The written british national corpus 2014 – design and comparability. *Text & Talk* **41**(5-6) (2021) 595–615
8. Sharoff, S.: Functional text dimensions for the annotation of web corpora. *Corpora* **13**(1) (2018) 65–95
9. Koppel, K., Kallas, J.: Eesti keele ühendkorpuste sari 2013–2021: mahukaim eesti-keelsete digitekstide kogu. *Eesti Rakenduslingvistika Ühingu aastaraamat* **18** (2022) 207–228

Utok: The Fast Rule-based Tokenizer

Pavel Rychlý and Samuel Špalek

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, Brno, Czech Republic
pary@fi.muni.cz

Abstract. Tokenization is one of the first processing steps in most natural language processing applications. The paper introduces a new tokenizer *Utok* which follows the *Unitok* tokenizer in the form of simplicity of configuration for different languages and is much faster in processing speed.

1 Introduction

Tokenization is a process of breaking down text data into minimal meaningful elements called tokens. That gives the machine ability to analyze each element in the context of other elements. It is one of the first tasks in the NLP text processing pipeline.

Most basic tokenizers use a simple idea of splitting text based on whitespace. That works well enough for some cases. The rest of the cases are more complicated, language-dependent, and require special treatment.

Examples of tokens we might want to recognize are:

- Words
- Numbers
- Dates
- Punctuation characters such as () . , [] { }
- Abbreviations (Etc., Jan., Mr., Gov.)
- Email
- URL
- SGML tags (HTML, XML, ...)

Tokens can also contain metadata. Most basic metadata would be a token position in the original text. Some tokenizers even categorize tokens into groups (e.g., POS tagging). We will focus on tokenizers that only separate tokens. Additional metadata can be added later upon further processing.

This paper introduces a new tokenizer: *utok*. It tries to solve the main problem of the *unitok* tokenizer: a slow processing of certain types of inputs. The paper also describes differences between *unitok* and *utok* tokenizers.

2 Tokenizers

2.1 Unitok

Unitok [4] is a tokenizer that takes text stream and outputs tokens separated by a newline. It is written in python language and uses the native python library for regular expressions (regexes). That results in not ideal performance. Configuration for a specific language is defined in the python code. Therefore configuration files are not universal and cannot be used out of the box in a different tokenizer.

Unitok supports many languages, such as Czech, Danish, Dutch, English, French, Finnish, German, Greek, Hindi, Italian, Maldivian, Spanish, Swedish, Yoruba. Other languages can be processed with default settings.

Glue tag In cases where there are two tokens next to each other, not separated with whitespace, a special glue tag `<g/>` is used.

An example of such case would be a dot at the end of a sentence. The last word in the sentence and the dot at the end is not separated with whitespace.

Unitok produces tokens: `last_word` `<g/>` `.`

Using the glue tag makes the tokenization reversible. It is possible to reconstruct original text from unitok output (same for utok).

Configuration files The unitok requires a configuration file for each language. It contains a list of regular expressions representing different types of tokens and the order of evaluation of that regular expressions. The form configuration file has a form of a Python source file, any Python constructs could be used.

Configuration of many languages uses advanced techniques like look-ahead to define some tokens. For example the following regular expression is used to describe order numbers in English:

```
ORDINAL = ur"""
(?<![-\w])
    ( \d+th | \d*(1st|2nd|3rd) )
(?<![-\w])
"""
```

Processing method The main part of the Unitok algorithm processes the input line by line and tries to find selected tokens in the line and splitting the line on that tokens. Then the smaller parts of the line are processed the same way in recurrence. That special tokens are selected in the order defined in the language configuration file.

2.2 Utok

Utok is a tokenizer created as a better version of unitok. It offers faster processing and a better, more universal configuration method.

For now, there is support for English and Czech language. Supporting a new language is pretty straightforward and basic configuration file (3.2) is good starting point.

Under the hood, utok is written in C++ and uses re2 regex library [2]. This library uses finite-state automaton and does not support advanced regex options that need to use functionality like backtracking (e.g., backreferences, look-around). That guarantees linear time complexity for every regex search.

Utok also supports "glue tag" and special "split" feature (3.1).

3 Utok configuration

Configuration is specified in a separate file. Each non-empty line contains a regex expression. All these regexes are parsed and concatenated into one big regex, that is then compiled using the re2 library. The configuration file supports comments using # symbol at the start of a line.

Each supported language has its own configuration file. Section 3.2 describes basic configuration file. It can be easily extended with language-specific tokens like abbreviations.

3.1 Split feature

Utok supports a special "split" feature. It is annotated with *SPLIT at the start of a line in a config file.

Utok goes through the text and tries to match one of the regexes from the config file. After a match is found, it tries to match regexes annotated with *SPLIT in order to split the token more and connect the parts with glue tag <g/>.

In English configuration it is used to separate apostrophe at the end of a word. For input "don't", the output tokens are

do	<g/>	n't
----	------	-----

This functionality produces significant slow down (see benchmarks 1). Utok performance is good, but it should be considered if the configuration for language needs to use this split feature.

3.2 Basic configuration

The example of what could be considered the default configuration for most languages is in Figure 1.

When creating a new language configuration, this configuration can be extended with language specific rules such as clitics, abbreviations, special word characters and emojis. The following example shows such extension for English which uses the SPLIT feature to separate English clitics as individual tokens.

```
*SPLIT (?i)(.+)('s|'re|'ve|'d|'m|'em|'ll|n't)
*SPLIT (?i)(can)(not)
```

```

# SGML tags
<[/?!]?[a-zA-Z] [-.:\w]*\s*/?.*>
<!--.*?-->

# XML entity
&(amp|lt|gt|quot|apos);

# URL
(https?|ftp|file)://\S*
\bwww\.[a-zA-Z0-9]+\.[a-zA-Z]{2,}(/S*)?
\b[a-zA-Z]([a-zA-Z0-9]+\.[a-zA-Z]{2,}(com|org|net|edu|gov|co|.uk)(/S*))?

# Email
\w[-'\.\w]*@([a-zA-Z0-9]+\.[a-zA-Z]{2,})

# Hashtag
[#@][a-zA-Z][a-zA-Z0-9]+

# Numbers
\d+([-+/.,\]\d+)*

# U.S.A.
([A-Z]\.)+\B

# Words
([\pL\pM\pN] | [\pL\pM\pN]) + (['\-\p{Pc}] [\pL\pM\pN] +)*

# Punctuation = any non-word, non-space
[?!]+
','
\.|\\*|:|=+
[^\pL\pM\pN\pZ]

```

Fig. 1: Basic configuration of utok

Table 1: Running time of Utok and Unitok on inputs of different size.

Program	English (17 MB)	English (91 MB)	Lorem ipsum with html (96 KB)	Czech (369 MB)
Utok	2.0s	16.7s	56.9ms	25.8s
Utok (no split)	1.0s	7.5s	49.9ms	25.8s
Unitok	12.3s	224.3s	426.3ms	621.6s

4 Benchmark

We have done basic evaluation of the speed of both Unitok and Utok on English and Czech texts. The results are summarized in Table 1. We can see the

slowdown of Utok with the SPLIT feature in the configuration file. The Czech configuration does not use the SPLIT feature.

5 Differences in output

During the creation of a utok configuration, the unitok was used as a baseline of good tokenization. However, there are cases in which it is harder to decide what behavior is better. In this section, we will explore a few differences found on English texts when comparing Unitok and the first version of Utok configuration.

The most frequent differences in our test data are the following:

1. Double symbol: for double symbols such as “// \$\$ %% --” unitok keeps them together `//`. On the other hand utok separates symbols with the glue tag `/` `<g/>` `/`.
2. When hash symbols is used like hashtag “#hashtag”, both tokenizers keep it as one token. Difference is when the hash symbols is between two words “word#hashtag”. Unitok separates the hash symbol with the glue tag on both sides `word` `<g/>` `#` `<g/>` `hashtag`. Utok keeps the hash symbol with the second word `word` `<g/>` `#hashtag`.
3. When talking about years in English we might encounter input “1980’s”. Unitok keeps it as one token. Utok separates it with the glue tag `1980` `<g/>` `'s`.
4. Apostrophe in non-usual places, for example foreign name “Tour de l’Aude”. Unitok separates apostrophe from both sides with glue tag `Tour` `de` `l` `<g/>` `'` `<g/>` `Aude`. Utok keeps the word together `Tour` `de` `l'Aude`.
5. Two dots at the end of a sentence in input “Title G. A. S..”. Unitok keeps the two dots together `Title` `G.` `A.` `S` `<g/>` `..`. Utok has more natural behavior `Title` `G.` `A.` `S.` `<g/>` `.`.

We can see that the differences are small and in many cases we think that Utok tokenization is better.

6 Conclusion

Utok is a tokenization tool that is built upon the previous work on unitok. The main advantage is speed and ease of configuration. Utok is an order of magnitude faster than Unitok. With even the basic configuration, it works well enough for unsupported languages.

In the future work we will adapt configuration files for all languages in Unitok to respective configuration in Utok. We will also experiment with other regular expression engines or implementation in other programming languages. We also plan to compare the tokenization of Utok with some highly used tokenizers for English ([1,3]).

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT/CLARIAH-CZ project (LM201810).

References



1. Altinok, D.: Mastering spaCy: An end-to-end practical guide to implementing NLP applications using the Python ecosystem. Packt Publishing Ltd (2021)
2. Google: Google/re2. <https://github.com/google/re2> (2022)
3. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations. pp. 55–60 (2014), <http://www.aclweb.org/anthology/P/P14/P14-5010>
4. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text tokenisation using unitok. In: Horák, A., Rychlý, P. (eds.) RASLAN 2014. pp. 71–75. Tribun EU, Brno, Czech Republic (2014)

Part IV

Semantics and Language Modelling

When Tesseract Meets PERO

Open-Source Optical Character Recognition of Medieval Texts

Vít Novotný  and Aleš Horák 

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
witiko@mail.muni.cz, hales@fi.muni.cz

Abstract. Conversion of scanned images to the text form, denoted as optical character recognition or OCR, for contemporary printed texts is widely considered a solved problem. However, the optical character recognition of early printed books and reprints of medieval texts remains an open challenge.

In our previous work, we developed an end-to-end image-to-text pipeline (via optical character recognition) for medieval texts, named AHISTO OCR, and we released it together with our test dataset under open licenses. However, the published system relied on the closed-source Google Vision AI service as one component, which made the experiments less reproducible. In this work, we replace Google Vision AI with an open-source OCR algorithm named PERO and we show that this not only makes the AHISTO OCR pipeline open, but also improves the performance of the system. We release the updated AHISTO OCR system and its test results again under open licenses.

Keywords: optical character recognition, OCR, medieval texts, AHISTO project

1 Introduction

In recent decades, public access to historical artifacts and documents has improved greatly via generally accessible photo banks and scanned sources such as the database of the Czech medieval sources online [7] provided by the Centre for Medieval Studies of the Czech Academy of Sciences. The utility of published documents further increases in case the data are not provided just in the image form but also with the recognized texts accessible to further content analysis via human search and automated natural language processing techniques.

The aim of the AHISTO project [1] is to make documents from the Hussite era (1419–1436) available to the general public through a web-hosted searchable portal and database. Although scanned images of modern letterpress reprints from the 19th and 20th century are available, accurate optical character recognition (OCR) algorithms are required to extract searchable text from the scanned images.

In our previous work [8,10], we have developed the AHISTO OCR pipeline for the image-to-text conversion of medieval texts in multilingual settings. We

have shown that the open-source Tesseract 4 ocr algorithm [14] was the second fastest and the most accurate among five different algorithms. [8] We have also shown that we can further improve the performance of the AHISTO ocr system by using Tesseract 4 for one-column pages and the closed-source Google Vision AI service [2] for two-column pages. [10] However, using a closed-source ocr service made our results difficult to investigate and reproduce.

In the current article, we present a new version of the AHISTO ocr pipeline where the Google Vision AI component is replaced with the open-source PERO ocr system [13]. We show that this not only makes the pipeline completely open, but the change also improves the performance of AHISTO ocr.

In Section 2, we introduce PERO ocr. In Section 3, we describe the experiments we have conducted and the dataset and measures that we used in our evaluation. In Section 4, we report the results of the evaluation. In Section 5, we summarize our contribution and outline the ideas for future work in the OCR of medieval texts.

2 Related Work

At ICDAR 2021, Michal Hradiš and his colleagues from the Brno University of Technology have presented different aspects of the new PERO ocr system [13,12]. Kodym and Hradiš [5] have introduced a page layout analysis algorithm for early modern and modern hand-written texts. Their algorithm achieved results comparable to state-of-the-art algorithms on a baseline detection task.

Kišš, Beneš, and Hradiš [4] have presented self-training and masked augmentation techniques for ocr algorithms. Their techniques led to significant improvements in performance on the optical character recognition of early modern and modern hand-written and printed texts.

Kohút and Hradiš [6] have showcased an adaptive instance normalization technique that can reconcile different transcription styles in ocr datasets produced by different annotators. Their technique makes it possible to use heterogeneous datasets to train ocr algorithms.

The PERO ocr system uses the above mentioned techniques for the optical character recognition of early modern and modern texts. The system is available as a web demo [13] and also as an open-source code at GitHub [12] with pre-trained models [3].

3 Methods

In our current work, we replace the Google Vision AI component of the AHISTO ocr system with two variants of the PERO ocr system: the web demo, with cloud-like service hosted at the Brno University of Technology, and the open-source code at GitHub with pre-trained models prepared for deployment at an independent server. In our previous work, we used the Google Vision AI model from October 2, 2020. To provide a fair comparison, we also report results with

a more recent model from August 11, 2022. As a baseline, we also report results for Google Vision AI and PERO OCR alone.

To evaluate the performance of the new AHISTO OCR version, we use the word error rate (WER) on the 120 human-annotated pages from the AHISTO dataset [11]. As in our previous work, we lower-cased and deaccented the texts in the dataset and in the predictions of our system to simulate a full-text search use case.

4 Results

In Table 1, we show that replacing Google Vision AI with PERO OCR improved the AHISTO OCR WER by 1.06%, to a very acceptable **2.08%**, even when the more recent Google Vision AI model from 2022 is used in the comparison.

The main complication in direct application of the two tested OCR systems was caused by the special format of two-column pages that appear in the scanned document collection with a non-negligible frequency. Whereas Google Vision AI achieved an extremely high WER of 78.35% on two-column pages with the earlier model from 2020, the more recent model is much better in analysing these page formats reaching an acceptable WER of 10.52%. To provide a fair comparison of PERO and Google Vision, we use AHISTO OCR with the 2022 Google Vision AI model in the evaluation.

The two variants of PERO OCR achieved different WER. This shows that the web demo of PERO OCR is not necessarily equivalent to the open-source code from GitHub with the published pre-trained models. To honor our intention to keep the AHISTO OCR system open-source, we employ the GitHub version of PERO OCR in the published AHISTO OCR pipeline.

Compared to the more recent Google Vision AI model alone, AHISTO OCR with PERO OCR improved WER by 1.54%. Compared to the web demo of PERO OCR alone, AHISTO OCR with PERO OCR improved WER by 3.97%.

Table 1: Word error rates (%) of Google Vision AI, PERO OCR, and AHISTO OCR evaluated on the AHISTO dataset [11]. For AHISTO OCR with Google Vision AI (the second column from the right), we report results with the more recent model from 2022-08-11. For AHISTO OCR with PERO OCR (the rightmost column), we report results with the open-source variant of PERO OCR available at GitHub and using the published pre-trained models. Best results in each row are **bold**.

	Google Vision AI		PERO OCR		AHISTO OCR	
	2020-10-02	2022-08-11	Demo	GitHub	with Google	with PERO
One column (103)	4.88%	3.79%	2.83%	2.08%	3.79%	2.08%
Two columns (17)	78.35%	10.52%	31.51%	49.38%	7.43%	9.93%
All pages (120)	16.23%	4.83%	7.26%	9.39%	4.35%	3.29%

5 Conclusion

In this work, we have shown that we can replace the closed-source Google Vision AI service with the open-source PERO OCR system to make the results of the image-to-text AHISTO OCR pipeline reproducible and also to improve the overall performance of the OCR system. We release the updated AHISTO OCR system [9] and its test results [11] under open licenses.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-CLARIAH-CZ project LM2018101.

References

1. Elbel, P., Novotný, R., Horák, A.: Accessible historical sources. Making medieval written documents available in the form of a contextual database, AHISTO, <https://nlp.fi.muni.cz/projects/ahisto>
2. Google: Vision ai, <https://cloud.google.com/vision>
3. Hradiš, M.: PERO EU-CZ print newspapers. Department of Computer Graphics and Multimedia, Faculty of Information Technology, Brno University of Technology, https://www.fit.vut.cz/~ihradis/pero/pero_eu_cz_print_newspapers_2020-10-09.tar.gz, [cit. 2022-08-09]
4. Kišš, M., Beneš, K., Hradiš, M.: AT-ST: Self-training adaptation strategy for OCR in domains with limited transcriptions. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR. pp. 463–477. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86337-1_31
5. Kodým, O., Hradiš, M.: Page layout analysis system for unconstrained historic documents. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR. pp. 492–506. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86331-9_32
6. Kohút, J., Hradiš, M.: TS-Net: OCR trained to switch between text transcription styles. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR. pp. 478–493. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86337-1_32
7. Novotný, R.: Czech medieval sources online. Centre for Medieval Studies, Institute of Philosophy, CAS CR, <https://sources.cms.flu.cas.cz>
8. Novotný, V.: When Tesseract does it alone: Optical character recognition of medieval texts. In: Horák, A., Rychlý, P., Rambousek, A. (eds.) RASLAN 2020. pp. 3–12 (2020)
9. Novotný, V.: ocr: An OCR engine for the AHISTO project. Faculty of Informatics, Masaryk University (Sep 2022), <https://gitlab.fi.muni.cz/nlp/ahisto-modules/ocr>
10. Novotný, V., Seidlová, K., Vrabcová, T., Horák, A.: When Tesseract brings friends: Optical character recognition of medieval texts. In: Horák, A., Rychlý, P., Rambousek, A. (eds.) RASLAN 2021. pp. 29–39 (2021)
11. Novotný, V., Seidlová, K., Vrabcová, T., Horák, A.: A human-annotated dataset of scanned images and OCR texts from medieval documents. Faculty of Informatics, Masaryk University (2022), <https://nlp.fi.muni.cz/projects/ahisto/ocr-dataset>, [cit. 2022-12-09]
12. PERO contributors: pero-ocr, <https://github.com/DCGM/pero-ocr>, commit 8908759
13. PERO contributors: PERO OCR demonstration application. Department of Computer Graphics and Multimedia, Faculty of Information Technology, Brno University of Technology (2022), <https://pero-ocr.fit.vutbr.cz/>, [cit. 2022-06-22]

14. Smith, R.: Tesseract blends old and new OCR technology. Tutorial at DAS (2016)

Medical Knowledge Resources for Text-Mining of Health Records in Czech, Polish, and Slovak

Křištof Anetta 

Natural Language Processing Centre, Faculty of Informatics, Masaryk University
Botanická 68a, Brno, Czech Republic
xanetta@fi.muni.cz

Abstract. Knowledge extraction from medical text in small languages like Czech, Polish or Slovak is challenging due to the insufficiency of language-specific medical resources (pretrained models, ontologies, dictionaries). This paper is a survey of noteworthy options for researchers targeting these languages, divided into two sections. First, since the UMLS Metathesaurus for English is by far the most extensive and detailed medical knowledge resource in Western medicine, appreciable results can be achieved by machine-translating the mined text to English – therefore, the relevant English components of UMLS are introduced. Second come the language-specific resources for each language, detailing the publishing institutions, current website locations, contents, and file formats. The contribution of this paper is in collecting and pre-screening widely disparate sources needed for successful medical knowledge extraction in Central European Slavic languages.

Keywords: EHR, electronic health records, healthcare text, UMLS, ICD-10, SNOMED CT, MedDRA, MeSH, NLP, natural language processing, Slavic languages, Polish, Czech, Slovak

1 Introduction

In small, low-resourced languages, producing a well-trained deep learning model for knowledge extraction from medical text is often impossible, the main culprits being limited data availability and even more limited capacity for expert annotation. Therefore, even as medical records in English have been revealing their secrets, small language medical corpora have remained a largely untapped resource of valuable information for both science and medical practice.

In small languages, vocabulary-based knowledge extraction methods are the keystone of all other efforts, identifying the literal occurrences of known medical concepts and, wherever possible, linking them to an existing ontology of medical knowledge.

Since it is highly labor-intensive to create new medical vocabularies that are properly linked to ontologies, the natural first step is to leverage existing resources, optimizing them for the task of medical knowledge extraction. This paper is a survey of major resources available for Czech, Polish, and Slovak and of the opportunities and limitations inherent in them.

If not stated otherwise, the focus is on resources that make it possible to locate medical concepts and entities with an unambiguous identifier within existing global coding/classification systems (ICD, ATC) or other major integrative initiatives (UMLS). This is to increase interoperability and enable international comparison in medical knowledge extraction.

2 UMLS: International resources for machine translation approaches

Even though the Unified Medical Language System [2], maintained by the United States National Library of Medicine, does have limited subset translations for individual Slavic languages, the full contents of the English UMLS Metathesaurus with millions of concepts and synonyms is the gold standard of vocabulary-based medical knowledge extraction (also used by Apache cTAKES [5], a leading clinical text analysis system) and its utility transcends the boundaries of English.

As can be seen in Table 1, the sheer advantage English has in concept counts and ready-made synonym permutations is so great that in some cases, instead of piecing the small-language-specific system together from comparatively tiny translated resources, it might well be rational to machine-translate all text to English and annotate it with the UMLS Metathesaurus.

Although a license is required to access the UMLS, there are several identity providers to choose from and its usage is free. Downloads¹ include the MRCONSO.RRF file, which contains all concepts, and is the key resource for string-based knowledge extraction (other UMLS files are mostly concerned with concept relationships, attributes, and indexes). Apart from the CUI (Concept Unique Identifier) which secures interconnection among the entire network of meanings, every concept is marked by its language and originating vocabulary, so text-mining researchers can easily filter it for the subset they are looking for.

All of the following resources for English are included in the UMLS release mentioned above.

¹ <https://nlm.nih.gov/research/umls/licensedcontent/umlsknowledgesources.html>

Table 1: UMLS Metathesaurus entry counts for relevant languages

Language	Entry count	Relative size
English	11,855,838	100%
Czech	212,304	1.8%
Polish	57,682	0.5%
Slovak	0	0%

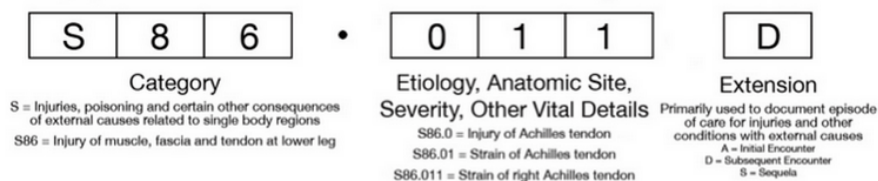


Fig. 1: ICD-10 code structure [6]

2.1 ICD

The International Classification of Diseases [9], maintained by the World Health Organization, is one of the most widely known classification systems in medicine. This paper deals with ICD-10, its tenth revision, as it has been in use for all or most of the time when electronic health records were being produced in the relevant countries.

ICD uses diagnostic codes (Figure 1) to classify diseases, symptoms, or abnormal findings. Depending on the granularity of representation, the ICD-10 may contain as many as 72,184 codes for different conditions [1]. Czech, Polish and Slovak each have a translation of the ICD-10.

The primary difficulty in using ICD-10 for knowledge extraction is the length of diagnosis names - unlike a traditional dictionary, full ICD-10 names rarely occur in the text and targeted approaches are necessary to atomize the long strings and locate their constituents. This problem will be addressed further in subsections devoted to individual languages.

2.2 SNOMED CT

The Systematized Nomenclature of Medicine Clinical Terms, or SNOMED CT [3], maintained by SNOMED International, a not-for-profit organization, is “the most comprehensive, multilingual clinical healthcare terminology in the world” [7]. As can be seen in Figure 2, it contains a much broader range of concepts than ICD-10, including diagnostic procedures, body structures or substances.

The 2020 international edition of SNOMED CT included 352,567 concepts [7]. Unfortunately, SNOMED CT has not been translated into any of the languages surveyed in this paper.

2.3 MedDRA

The Medical Dictionary for Regulatory Activities (MedDRA), maintained by the International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (ICH), is an international medical terminology dictionary-thesaurus translated into several languages including Czech. Its

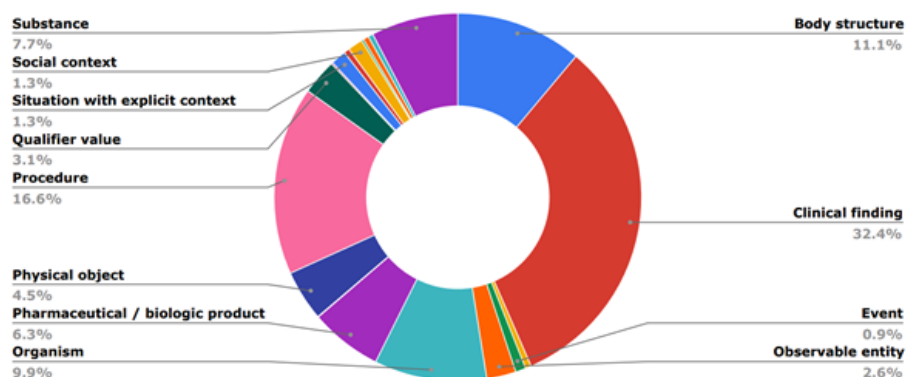


Fig. 2: Breakdown of concepts in SNOMED CT[7]

primary use is in regulatory communication in the biopharmaceutical industry, describing concepts in the clinical research of medicinal products, including adverse events. There are 115,479 English MedDRA concepts in the UMLS. They are organized as a five-level hierarchy:

- SOC (System Organ Class)
- HLT (High-Level Group Term)
- HLT (High-Level Term)
- PT (Preferred Term)
- LLT (Lowest Level Term)

2.4 MeSH

Medical Subject Headings (MeSH) [4] is a “controlled and hierarchically-organized vocabulary” [8] produced by the United States National Library of Medicine. Its primary use is indexing of journal articles and books in the life sciences, so it can be expected to have less utility in discovering detailed medical information about patients than the previously mentioned vocabularies.

From the languages relevant for this paper, MeSH has been translated into Czech and Polish.

3 Czech

3.1 MedDRA

Unlike Polish or Slovak, Czech provides the option of a MedDRA translation included in the UMLS release listed above. It contains 76,255 unique strings in an interconnected and hierarchical collection of 111,573 entries, the average length of unique entries being 3.74 words. This is relatively high for string lookup, as the probability of unmodified occurrence decreases sharply above 3 words and

the Czech MedDRA contains 31,083 unique entries longer than 3 words. Table 2 shows entry counts according to hierarchy level.

3.2 MeSH

Like MedDRA, the Czech MeSH is also included in the complete UMLS release. Czech MeSH is maintained by the Czech National Medical Library. It contains 100,731 entries, 100,618 of them unique, and the average length is 2.28 words, which is fortunate for exact string lookup.

3.3 Registered drug list

The Czech State Institute for Drug Control maintains several drug-related databases². The most relevant resource is the current version of the list of registered drugs³. The ZIP file contains several CSV files, of which the most important one, currently named `dlp_lecivepripravky.csv`, contains structured information for each drug organized into 41 columns including the ATC code, which can serve as an international identifier.

There are 63,066 entries in the drug list including different strengths and packagings of the same drug, which boils down to just over 7,500 unique strings.

3.4 ICD-10 translation

The Institute of Health Information and Statistics of the Czech Republic publishes⁴ the Czech translation of the ICD-10, referred to as MKN-10.

The files available for download are in a ZIP file referred to as “CSV strukturované podklady” and they are divided into UTF-8 and Windows 1250 versions. The main file (`01_MKN10_5E_2022_cp_w1250.csv` or `01_MKN10_5E_2022_utf8.csv` in the most recent version) contains the concepts ordered by code, starting with A00.

² <https://opendata.sukl.cz>

³ <https://opendata.sukl.cz/?q=katalog/database-lecivych-pripravku-dlp>

⁴ <https://uzis.cz/index.php?pg=registry-sber-dat--klasifikace--mezinarodni-klasifikace-nemoci-mkn-10#publikace>

Table 2: Czech MedDRA structure

Hierarchy level	Hierarchy code in UMLS release	Concept count
1 - SOC	OS	27
2 - HLG	HG	337
3 - HLT	HT	1,737
4 - PT	PT	25,077
5 - LLT	LLT, OL	84,139

However, identifying ICD-10 concepts in text is a major challenge due to the long, detailed names of the conditions, extremely unlikely to be found verbatim. As can be seen in Table 3, the average diagnosis name length is 4.65 words, with 19,007 diagnoses (48.6%) of 5 words and more. However, the release of the Czech translation of ICD-10 contains separate files (CSV files with names containing “Abecední seznam”) which contain an alphabetically ordered index of concepts split into a hierarchy. Table 4 demonstrates this, showing also that corresponding ICD codes are present in no particular order. With some rule-based processing that removes redundant or impractical text (such as references like “viz” and the contents of parentheses), lemmatizes words, and potentially splits subconcepts into subsubconcepts based on commas, the resulting strings end up much shorter and much more likely to be found in texts on their own. Table 3 shows the dramatic decrease of string length with the index file and with further automated optimization. Together with a cluster searching method that finds collocated subconcepts, this makes the ICD system much better suited for lookup in messy text where diagnoses hide in jumbled or incomplete forms.

Table 3: Czech ICD-10 translation file usability

File	Average item length (words)
Main file ordered by code	4.65
Hierarchical index	2.75
Optimized hierarchical index	1.57

Table 4: Czech ICD-10 alphabetical hierarchy example

Level 1	Level 2	Level 3	Level 4	ICD-10 code
Absorpce				
	bílkovin, porucha			K90.4
	dusíkatých látek – viz Uremie			
	glycidů, sacharidů, porucha			K90.4
	chemikálie			T65.9
		transplacentární (plodem nebo novorozencem)		P04.9
			látky z výživy	P04.5
			látky z životního prostředí	P04.6

4 Polish

4.1 MeSH

The major Polish subset of the UMLS Metathesaurus (available in the UMLS release mentioned above) is MeSH, published by the Polish Central Medical Library. It is smaller than the Czech one with 53,542 entries and an average length of 2.29 words, comparatively suitable for string search.

4.2 Databases published by the Ministry of Health

The Polish Ministry of Health publishes several relevant medical concept databases at ⁵, including the list of registered drugs and the Polish translation of ICD-10.

Registered drug list The list of registered drugs is available as a XLSX⁶ or a CSV⁷ file. There are 24 columns containing information about each entry, including extraction-relevant strings like product name, active ingredient, strength, packaging variants, and the internationally recognized ATC code.

There are 22,231 entries in the file, boiling down to just over 20,000 unique names.

The drug names included in this list only need minor automated editing (such as the optional isolation of name alone from manufacturer names, e. g. “GSK” in “Nitrazepam GSK”, or quantities, e. g. “150 mg” in “Ranisan 150 mg”) and the resulting vocabulary becomes a highly accurate tool for locating drug name mentions.

ICD-10 translation The Polish translation of ICD-10 can be downloaded⁸ as an XML file with hierarchically organized nodes based on the granularity of representation (Figure 3).

11,314 diagnosis names can be extracted from this file and the average length is 5.33 words, which indicates that most of the strings are not ready to be searched for in medical text. An alphabetically sorted, hierarchical file is not available for Polish, so other automated techniques (such as word separation and cluster search) have to be used to locate ICD-10 concepts in real-world use.

⁵ <https://rejestrymedyczne.ezdrowie.gov.pl>

⁶ <https://rejestrymedyczne.ezdrowie.gov.pl/api/rpl/medicinal-products/public-pl-report/get-xlsx>

⁷ <https://rejestrymedyczne.ezdrowie.gov.pl/api/rpl/medicinal-products/public-pl-report/get-csv>

⁸ section “Pliki do pobrania” at <https://rsk3.ezdrowie.gov.pl/resource/structure/icd10/00CD10/011/url>

```

<node code="A50-A64">
  <name>Zakażenia przenoszone głównie drogą płciową</name>
  <attributes>
    <attribute name="EN">Infections with a predominantly sexual mode of transmission</attribute>
  </attributes>
  [...]
  <nodes>
    <node code="A50">
      <name>Kłta wrodzona</name>
      <attributes>
        <attribute name="EN">Congenital syphilis</attribute>
      </attributes>
      <nodes>
        <node code="A50.0">
          <name>Kłta wrodzona wczesna objawowa</name>
          <attributes>
            <attribute name="EN">Early congenital syphilis, symptomatic</attribute>
          </attributes>

```

Fig. 3: Polish ICD-10 translation data example

5 Slovak

5.1 Registered drug list

The Slovak State Institute for Drug Control maintains several drug-related databases at ⁹, including an up-to-date list of drugs registered in Slovakia¹⁰. It can be downloaded as an easily processable XLS file with multiple columns where the structure is similar to that of the Czech registered drug list, including an ATC code which can be used to link the specific drug to all its associated concepts within the UMLS.

There are 51,314 entries in the file, out of which 9,090 are unique strings.

5.2 ICD-10 translation

The Slovak National Health Information Center publishes the Slovak translation of ICD-10, referred to as MKCH-10¹¹. Like Polish, there is only one file where diagnoses are ordered by code and no granularized file is available.

The multi-sheet XLS file has 20,029 lines of diagnoses and diagnostic categories, with an average length of 7.68 words, making further automated subdivision necessary for successful string search to be feasible.

6 Conclusion

The available resources for Czech, Polish and Slovak indicate a clear direction for medical knowledge extraction, but they are far from making it straightforward, either due to machine translation issues (when using English resources) or as a result of differences between concepts in dictionaries and actual strings found in medical text. Some information, such as drug names, are easy to extract thanks

⁹ www.sukl.sk/verejne/

¹⁰ www.sukl.sk/verejne/Zoznam_liekov/zoznam_liekov.zip

¹¹ www.nczisk.sk/Standardy-v-zdravotnictve/Pages/Medzinarodna-klasifikacia-chorob-MKCH-10.aspx

to them being mostly one- or two-word strings occurring together. Others, such as ICD diagnoses, are very difficult to find and require dedicated methods of preprocessing of the longer names into subconcepts and special ways of searching for their collocation.

However, if medical informaticians focused on Slavic languages coordinate their efforts, develop pipelines that transform available resources into vocabularies usable for string search, and publish the newly created resources to facilitate an accumulative effect, the resulting systems have the potential to be a major leap in the area of Slavic medical text processing. Poland, the Czech republic, and Slovakia have a combined population of almost 55 million, and this would open up decades of this whole region's health documentation, public or private, to automated analysis, semantic filtering, and statistical research.

Acknowledgements The research in this paper was supported by the Internal Grant Agency of Masaryk University, project MUNI/IGA/1326/2021: *New Horizons of Electronic Health Record Analysis using Deep Learning*, and it was partially carried out within the project MUNI/G/1763/2020: *Alcope - AI support for Clinical Oncology and Patient Empowerment*.

References

1. American Academy of Professional Coders: All about ICD-10 (May 2021), <https://www.aapc.com/icd-10/>
2. Bodenreider, O.: The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research* **32**(suppl_1), D267–D270 (01 2004). <https://doi.org/10.1093/nar/gkh061>, <https://doi.org/10.1093/nar/gkh061>
3. Donnelly, K., et al.: SNOMED-CT: The advanced terminology and coding system for eHealth. *Studies in health technology and informatics* **121**, 279 (2006)
4. Lipscomb, C.E.: Medical subject headings (MeSH). *Bulletin of the Medical Library Association* **88**(3), 265 (2000)
5. Savova, G.K., Masanz, J.J., Ogren, P.V., Zheng, J., Sohn, S., Kipper-Schuler, K.C., Chute, C.G.: Mayo clinical text analysis and knowledge extraction system (cTAKES): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association* **17**(5), 507–513 (2010)
6. Smith, J.: Understanding the ICD-10 code structure (Mar 2022), <https://www.webpt.com/blog/understanding-icd-10-code-structure/>
7. SNOMED International: 5-step briefing, <https://www.snomed.org/snomed-ct/five-step-briefing>
8. U.S. National Library of Medicine: Medical subject headings - home page, <https://www.nlm.nih.gov/mesh/meshhome.html>
9. World Health Organization: ICD-10 : International statistical classification of diseases and related health problems : Tenth revision (2004)

Secondary Prepositions with Causal Meaning in Russian

Victor Zakharov¹, Kirill Boyarsky², Eugeny Kanevsky³, and Anastasia Kozlova¹

¹ Saint Petersburg University
Universitetskaya emb. 7-9
199034 Saint Petersburg, Russia
v.zakharov@spbu.ru, stasia.kozlova@gmail.com

² ITMO University
Kronverkskiy av. 49
197101 Saint Petersburg, Russia
boyarin9@yandex.ru

³ Institute of Regional Economics Problems RAS
Serpukhovskaya St. 38
190013 Saint Petersburg, Russia
eak300@mail.ru

Abstract. The present study deals with Russian secondary prepositions, primarily focusing on multiwords. Secondary prepositions are units motivated by content words (nouns, adverbs, verbs), which may be combined with primary prepositions to form multiword prepositions (MWPs). Multiword prepositions perform the grammatical function of a preposition in a certain position of a syntactic structure in some contexts and can be a free word combination in others. This paper is devoted to analysis of the use of secondary multiword prepositions with causal meaning. We analyze the repertoire of Russian MWP causal prepositions and describe their statistical representation in corpora.

Keywords: Russian language, secondary prepositions, multiword prepositions, causal meaning, corpus statistics, parsing

1 Introduction

This study is part of a project whose goal is to create the first corpus semantic-grammatical description of Russian prepositional constructions. The Russian linguistic tradition implies the division of the class of prepositions into primary and secondary in origin, as well as into simple (single-word) and complex (multiword) units in structure. While the subclass of primary prepositions is relatively well studied and well documented, secondary prepositions have not received the same attention in the linguistic literature, despite making up the majority of prepositions as a class. The preposition is a common part of speech in many languages. It has been established that prepositions in Russian make up, on average, 10% of all tokens in each text [1].

The vague and complex semantics of prepositions underlies much of the debate about the nature of prepositions. Primary prepositions are very ambiguous. For example, the Russian preposition *в* ‘in’ has 23 meanings in the Dictionary of the Russian Language [2]. In other cases, the primary preposition is part of the secondary preposition. In total, there are several hundred secondary prepositions in the Russian language. Most often they can be considered as synonyms for primary ones.

We understand a prepositional meaning as a relation that occurs in prepositional constructions, where it should be considered as a special type of relationship between meaningful words. We consider this concept as a semi-grammatical component of the language, linking fuzzy lexico-semantic classes of words. These relationships are established by a combination of a specific preposition, the semantic type of the lexeme that attaches the prepositional construction, and the case and the semantic class of the dependent word. Therefore, to describe and analyse causal prepositions, we provide parsing and identify the semantic classes of dependent words.

2 Related Work

Perhaps the most well-structured inventory of Russian secondary prepositions can be found in the Russian Grammar [3]. It is noted by the author(s) that a lot of the units listed are entities of uncertain part-of-speech status due to their preserved ability to include determiners and combine selectively with the other parts of the potential prepositional phrase [3, §1661]. The Explanatory Dictionary of Functional Parts of Speech of the Russian Language [4] contains less than 300 secondary prepositions. Much fewer, just 157, are found in the Explanatory Dictionary of Combinations Equivalent to a Word [5].

When speaking about causal relationship, a lot of studies are devoted to this type of relation ([6,7,8] and many others) and only several studies deal with causal prepositions [9,10,11]. However, these studies are not based on corpora. Our study relies on statistics on the use of these prepositions in large text material and on syntactic parsing.

3 Secondary Multiword Prepositions with Causal Meaning

Secondary prepositions are words and phrases that have assumed the function of a preposition. Structurally, these units can be divided into simple and complex (multiword) ones. Simple secondary prepositions are usually fully homonymous with some word form of their motivating content word or a different part of speech sharing the same root. The same words and word units may perform as prepositions as well as other parts of speech (e.g.: *силами* ‘by force of’ – preposition, noun, *снаружи* ‘outside’ – preposition, adverb, *исключая* ‘excluding’ – preposition, verb (participle)).

Multiword prepositions (MWP) make up a large part of secondary prepositions. Structurally speaking, a multiword preposition is a combination of a

content word and one or two simple adpositions. MWPs can be divided into nominal, adverbial or verbal units based on the part of speech of the motivating content word. Most MWPs contain only one adposition preceding or following the content word (e.g.: *рядом с* 'close to', *в результате* 'as a result'), but some include two adpositions enclosing the content element (e.g.: *в соответствии с* 'in accordance with', *по направлению к* 'toward, in the direction of'). The most commonly observed structural patterns of MWPs are Prep+N, Prep+N+Prep and Adv+Prep, where Prep stands for preposition, N for noun, Adv for adverb. Much like simple secondary prepositions, multiword prepositional units perform as prepositions in some contexts and as free word combination in others (e.g., preposition + noun: *в форме* 'in the form of', conjunction + preposition: *что до* 'as for'; verb + preposition: *начиная с* 'starting with').

As a rule, the distinction between these ambiguous entities is outlined neither in grammar books nor in dictionaries. The great variety of MWPs on all language levels implies the necessity of an in-depth analysis of their common features. In other words, we need to understand, firstly, what unites such diverse entities in order to be able to discern free combinations from MWPs.

As has already been stated, prepositional multiword entities do not always function unambiguously as MWPs. Although our current paper is devoted to causal prepositions, in order to define limits of all MWPs, we have formulated the following preliminary list of the main characteristic features of multiword prepositions:

- MWP performs the grammatical function of a preposition in a certain syntactic position as part of a prepositional phrase; that is, it governs a noun or a nominalised word (sometimes an infinitive).
- MWP inherits the semantics of the notion word (noun, verb); it derives from as well as its valency (*на основе* 'on the grounds of' – *основа чего?* 'the grounds of what?'; *в зависимости от* 'depending on' – *зависеть от чего?* 'to depend on what?'; *с целью* 'with the aim to' – *цель что сделать?* 'aim to do what?').
- As a rule, it contains one or two primary prepositions.
- Its nominal components tend to have abstract semantics.
- It has a relatively high frequency among multiword units of the same structural type.
- It is idiomatised, i.e., its nominal component loses its lexical meaning to an extent (which is why MWPs are sometimes called "prepositional idioms").
- The grammatical number of the noun cannot be changed (it is either singular or plural).
- It has a primary preposition as a synonym.
- In most cases, it does not allow for insertion or separation (as a rule, the noun cannot have a possessive or adjectival determiner).
- All of these features are characterised by significant statistical regularity.

4 Material

Causal preposition meaning is the meaning of constructions where the prepositional group indicates the cause of an action or the influencing factor. The word 'cause' is the basic term used to interpret the whole lexical composition, which is associated with the category of determining the cause. Dictionaries of a language divide two meanings of this word: 1) cause as a phenomenon that inadvertently causes another phenomenon, ontological cause; 2) cause as a basis, precondition for the realization of an event, action, i.e., subjective, explainable cause. The meaning of prepositions is based on that description. Among ways to express causal relations, the most common are prepositional-case forms and complex sentences with a subordinate causal part, and most of causal conjunctions come from prepositions.

Causal relationships can be expressed in Russian by some primary prepositions and a large number of secondary ones. They can enter into connection with nouns and pronouns in genitive, dative, accusative and instrumental. The lists of causal prepositions differ in different sources. According to our analysis, the list is as follows: *за, из, из-за, на, от, по, под, после, при, с, через* (primary prepositions), *благодаря* 'thanks to', *в зависимости от* 'depending on', *в ответ на* 'in response to', *в результате* 'as a result of', *в свете* 'in light of', *в связи с* 'due to', *в силу* 'by force of', *за счёт* 'on account of', *исходя из* 'drawing from', *на основании* 'on the basis of', *на основе* 'based on', *на почве* 'on the ground of', *по причине* 'because of, for the reason of' (secondary prepositions). (Here we provide English equivalents only for the secondary prepositions because the meanings of the primary ones are highly context-dependent). These prepositions form clusters of synonymy. Different prepositions can express the same meanings and grammatical relations when used in the same phrases. In the sentences *Он не пришел по причине болезни – Он не пришел из-за болезни – Он не пришел вследствие болезни* ('He did not come because of the disease') it is possible to interpret the prepositions as synonyms.

Our selection consists of 13 multiword preposition candidates that have been observed to express causal or causal-adjacent relations:

- *в зависимости от* 'depending on'
- *в ответ на* 'in response to'
- *в преддверии* 'on the eve of, at the forefront of'
- *в результате* 'as a result of'
- *в свете* 'in light of'
- *в связи* 'due to'
- *в силу* 'by force of'
- *за счёт* 'on account of'
- *исходя из* 'drawing from'
- *на основании* 'on the basis of'
- *на основе* 'based on'
- *на почве* 'on the ground of'
- *по причине* 'because of, for the reason of'

The results of the statistical analysis presented in this article have been acquired on the Russian National Corpus (RNC, www.ruscorpora.ru). This corpus was chosen due to its considerable size (about 375 million tokens). For parsing and manual (intelligent) analysis, random samples of 500 sentences were taken from the RNC for each preposition.

5 Results

5.1 Structural Features of Causative MWPs

The first set of features to observe is the structure of the MWPs in question. 10 out of 13 units are bigrams of a content word and a simple adposition, which appears to be the most typical MWP structure. 9 of the bigram units follow the structural pattern of Prep+Noun, one has the less common pattern of Verb+Prep. The remaining 3 out of 13 units are trigrams consisting of a noun between two adpositions.

As has been noted by us in [12] the three simple adpositions most commonly used as elements of multiword prepositions are *в*, *на*, *по*. Out of the 13 items under current study, 7 contain the preposition *в*, 4 contain *на*, 1 contains *по*.

Most of the content words in the causative MWPs refer to the two nodes of causal relations: the reason (*основание* 'foundation', *основа* 'base', *почва* 'ground', *причина* 'reason') and the effect (*ответ* 'response', *результат* 'result'), as well as the relation itself (*зависимость* 'dependency', *связь* 'connection', *сила* 'force'). The semantics of the motivating content words corresponds with the observed tendency of MWP component nouns to lean towards abstraction.

5.2 Statistical Analysis of Causative MWPs

Another point of interest is the use (frequency) of the content (base) words as multiword unit (MWU) components in comparison to their general corpus frequency. The table below demonstrates the frequencies of the content words in question as well as the frequencies of the multiword unit themselves (Table 1).

As demonstrated by the table, most of the content words retain their relative independence as they are not bound to the other parts of the MWUs. Only two of them, *в преддверии* and *исходя из*, appears to be a set phrase in which the motivating word is not almost used without the corresponding preposition.

To gain a clearer understanding of how prepositional units function as MWPs or free combinations we have studied the use of the causative prepositional units in the corpus. The results are presented in Table 2.

The results show that these word combinations are mostly used as prepositions and not free word combinations. MWPs which demonstrate a relatively low percentage of prepositional sense have a strong sememe as the base word that retains its original meaning even as part of a preposition. For example, in the preposition *на основе* 'on the basis of', the base word 'base' tends to retain approximately the same meaning both in the preposition and outside it.

Table 1: Frequency counts of MWUs in relation to base word frequencies in RNC

Base word	Multiword unit (MWU)	MWU, count	Base word, count	MWU/base word, %
<i>Исходя</i> ⁴	<i>Исходя из(о)</i>	5314	5788	91,8
<i>Преддверие</i> ⁵	<i>В преддверии</i>	1134	1373	82,6
<i>Зависимость</i>	<i>В зависимости от(о)</i>	9859	21604	45,6
<i>Результат</i>	<i>В результате</i>	29745	93454	31,8
<i>Счёт</i>	<i>За счёт/ет</i>	16113	52693	30,6
<i>Связь</i>	<i>В связи с(о)</i>	23496	96248	24,4
<i>Основа</i>	<i>На основе</i>	12764	52299	24,4
<i>Основание</i>	<i>На основании</i>	11296	48353	23,4
<i>Почва</i>	<i>На почве</i>	2929	25772	11,4
<i>Сила</i>	<i>В силу</i>	14884	266166	9,0
<i>Ответ</i>	<i>В ответ на</i>	6723	77869	8,6
<i>Причина</i>	<i>По причине</i>	4557	81422	5,6
<i>Свет</i>	<i>В свете</i>	5727	156587	3,7

Table 2: Percentage of prepositional use of MWP candidates in Russian National Corpus

MWP	% of prepositional use
<i>Исходя из(о)</i>	100,0
<i>В результате</i>	100,0
<i>По причине</i>	100,0
<i>В зависимости от(о)</i>	99,8
<i>В связи с(о)</i>	99,4
<i>На основании</i>	99,8
<i>В ответ на</i>	99,8
<i>В преддверии</i>	98,4
<i>За счёт/ет</i>	96,2
<i>На почве</i>	96,6
<i>В силу</i>	83,6
<i>На основе</i>	79,0
<i>В свете</i>	49,2

For example, *на основе* was found to be a free word combination in contexts referring to the physical basis of an entity, e.g. [X] “*на основе гиалуроновой кислоты*” (‘hyaluronic acid-based [X]’); similarly, word combination *в свете* ‘in light of’ was used literally in contexts where the governee belonged to the semantic class of objects capable of emanating light, e.g. “*в свете заходящего солнца*” (‘in the light of the setting sun’). The MWP candidate *в силу* ‘by force of, due to’ was found to be used occasionally as a free combination ([*верить*] *в силу* ‘[believe] in the force’) and as part of an adverbial idiom ([*вступить*] *в силу* ‘come into power’), which led to the relatively lower observed percentage of its prepositional use as well.

A special point of interest is the separability of MWPs, that is, the allowance for modifier insertion into the MWP structure. To study this phenomenon, we

have examined context samples of some causative MWP candidates with and without content word modifiers. Overall, our presupposition that insertion is atypical for MWPs has been proven true. Since most of the content words in our MWP candidate selection are nouns, it was primarily adjectival modifiers that were found splitting the original prepositional unit structure. As it was said, the nominal component of MWPs loses often its lexical meaning. When modifier insertion takes place, the semantic weight of the whole construction figuratively shifts back to the modified noun, which retains its original lexical meaning. Therefore, the resulting structure can no longer perform the prepositional function and can only be regarded as a free word combination. Some nominal causative MWPs do not seem to lose their function in the case of anaphoric use of personal pronouns, such as *на его почве* 'on his [its] ground', *в её преддверии* 'on her [its] eve'.

The meaning of a secondary preposition depends sometimes on the meaning of semantics of a governing word and of a governee (dependent word). We have obtained governee lists (see Table 3) for each of the prepositional units in question by the SemSin parser [13], which builds a dependency tree for each sentence and detects types of relations between its nodes. The parser relies on a semantic-syntactic dictionary and a classifier, both of which are extensions of the semantic dictionary by V.A. Tuzov.

The most frequently occurring dependent words for each set phrase often quite clearly illustrate the meaning of construction, whether it is a preposition or not. It is also noticeable that for the preposition with the lowest percentage of prepositional meaning, *в свете* 'in the light of', one of the most frequent dependent words is 'campfire': it becomes obvious that this phrase is often used as free combination. Dependent word classes influence the fact of whether the word is used as a preposition component within the MWP: the more dynamic the meaning of the semantic class is, the more likely it is to be non-prepositional.

6 Conclusion

To conclude, it can be noted that analyzed MWUs quite often perform the function of prepositions (MWPs). We can conclude also that causative MWPs generally do not allow for insertion (modification of the content component) except when the modifier is a personal pronoun modifying the nominal component or a particle modifying the verbal component. Whether this rule applies to the entirety of the MWP class is subject to further investigation.

The most frequent governors in prepositional usage cases belong to the group of verbs and verbal nouns expressing change of state, e.g., *получить* 'receive', *возникать* 'appear', *образоваться* 'form', or expressing difference, e.g., *меняться* 'change', *варьироваться* 'vary', *отличаться* 'differ'; for the preposition *в зависимости* 'depending on'. For the preposition *в силу* 'due to, by force of' the governors were useful in identifying free usage cases, e.g., *вступление* 'entry', *вступить* 'come', *верить* 'believe' [in(to) (the) power]. Inversely, the homonymy resolution of the MWP candidates *в свете*, *в преддверии* was more

Table 3: Most frequent governees and governees semantic classes in causal MWP constructions

MWP	The most frequent governees	The most frequent semantic classes
<i>В преддверии</i>	выборов 'elections' года 'of the year' юбилея 'anniversary'	Обладание 'Possession' Конкретное время 'Specific_time' Занятие 'Occupation'
<i>В зависимости от(о) того</i>	'that' условий 'conditions' типа 'like'	Спецкласс_для_TO_и_ЭТО 'Special_class_for TO_and_ЭТО' Событие 'Event'
<i>Исходя из(о)</i>	этого 'that' того 'that' опыта 'experience'	Качество 'Quality' Сообщение 'Message' Этот-Другой 'This_Other'
<i>За счёт/ет</i>	средств 'means' использования 'of using' того 'that'	Дух 'Spirit' Деньги 'Money' Изменение 'Change'
<i>В связи с(о)</i>	этим 'by this' развитием 'development' делом 'case'	Занятие 'Occupation' Этот-Другой 'This_Other' Событие 'Event'
<i>На основе</i>	анализа 'analyses' данных 'data' опыта 'experience'	Дело 'Business' Науки 'Science' Сообщение 'Message'
<i>На основании</i>	данных 'data' того 'that' анализа 'analysis'	Документы 'Documents' Сообщение 'Message' Нечто 'Something'
<i>В результате</i>	которой 'which' работы 'work' деятельности 'activity'	Документы 'Documents' Изменение 'Change' Сообщение 'Message'
<i>По причине</i>	того 'that' отсутствия 'absence' болезни 'illness'	Нечто 'Something' Событие 'Event' Дух 'Spirit'
<i>В ответ на</i>	вопрос 'question' это 'this' просьбу 'request'	Душа 'Soul' Сообщение 'Message' Этот-Другой 'This_Other'
<i>На почве</i>	любви 'love' ревности 'jealousy' отношений 'relations'	Душа 'Soul' Дух 'Spirit' Борьба 'Struggle'
<i>В свете</i>	событий 'events' сказанного 'said' костра 'campfire'	Событие 'Event' Сообщение 'Message' Стихия 'lement'
<i>В силу</i>	причин 'reasons' того 'that' обстоятельств 'circumstances'	Событие 'Event' Причина 'Cause' Дух 'Spirit'

successful in the presence of their governees. As such, contexts with the governess *фары* 'headlights', *фонари* 'streetlamps', *луна* 'the moon' for the preposi-

tion *в свете* 'in light of' and *пот* 'mouth', *влагалище* 'vagina' for the preposition *в передверии* 'at the forefront of' were found to be free word combinations. Another interesting observation is that high number of the causative prepositional units were found to take the initial position in a sentence, or a clause as evidenced by the inclusion of punctuation marks and conjunctions.

In the future, we plan to describe MWP's with other meanings.

References

1. Lyashevskaya, O.N., Sharov, S.A.: Frequency Dictionary of the Modern Russian Language (on the Materials of the National Corpus of the Russian Language). Azbukovnik, Moscow (2009).
2. Dictionary of the Russian Language, 4 volumes. 3rd edn. Russkiy yazyk, Moscow (1988).
3. Shvedova, N.Ju.: Russian Grammar. Vol. 1: Phonetics. Phonology. Word Stress. Intonation. Word Formation. Morphology. (in Rus). Nauka, Moscow (1980).
4. Efremova, T.F.: Explanatory dictionary of functional parts of speech of the Russian language. AST, Moscow (2004).
5. Rogozhnikova, R.P.: Explanatory dictionary of combinations equivalent to a word: Approx. 1500 fixed phrases of the Russian language. AST, Moscow (2003).
6. Diessel, H., Hetterle, K.: Causal clauses: A crosslinguistic investigation of their structure, meaning, and use. In: P. Siemund (ed.). Linguistic Universals and Language Variation. Berlin, Boston: De Gruyter Mouton, p. 21 – 52 (2011).
7. Kroupová, Libuše. Vztah významu gramatického a lexikálního u předložek. In: Slovo a slovesnost, 1980, roč. 41, č. 1, s. 49 – 52 (1980).
8. Vsevolodova, Maya – Yashchenko, Tatyana: Causal relationships in modern Russian. Moscow: Russkiy yazyk. 208 p. (in Russian) (1988).
9. Levontina, Irina: <Causal prepositions>. In: New explanatory dictionary of synonyms of the Russian language. Moscow, p. 144 – 152. (in Russian) (1997).
10. Iordanskaya, Lidiya – Melchuk, Igor: On the semantics of Russian causal prepositions. In: Linguistic Journal. No. 2, p. 162–211. (in Russian) (1996).
11. Luraghi, Silvia. Prepositions in Cause expressions. In: Papers on grammar, vol. 12, No. 2, p. 609–619 (2005).
12. Zakharov, V., Azarova, I.: Semantic structure of Russian prepositional constructions. In: K. Ekstein, V. Matousek (eds.). Lecture Notes in Computer Science, vol. 11697 LNAI (Text, Speech, and Dialogue - 22th International Conference, TSD 2019 Proceedings), pp. 224-235. Springer International Publishing AG (2019).
13. Boyarsky K., Kanevsky Ye. Semantic-syntactic parser SemSin [Semantiko-sintaksicheskii parser SemSin]. In: Scientific and technical bulletin of information technologies, mechanics and optics [Nauchno-tehnicheskiiy vestnik informacionnykh tekhnologiy, mekhaniki i optiki]. Vol.15. No. 5, pp. 869-876 (2015).

Towards General Document Understanding through Question Answering

Šárka Ščavnická , Michal Štefánik , Marek Kadlčík , Martin Geletka ,
and Petr Sojka 

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
527352@mail.muni.cz

Abstract. Document Visual Question Answering is a relatively new extension of Visual Question Answering. The aim is to understand the documents and to be able to obtain information that corresponds to the question that was asked. This proposition aims to approach the problem of the lack of datasets and a model for Slavic languages. Therefore we would like to create a model and dataset for Document VQA suitable for the non-English language. This paper overviews the field of Question Answering and also describes the first Czech Document VQA dataset and model.

Keywords: Question Answering, Visual Question Answering, Document Visual Question Answering

1 Introduction

Document processing and analysis is a rapidly growing field. It applies not only to analysts who try to obtain and analyze critical information. It is also used in economic sectors, where they speed up the processing of legal documents and invoices. Several new works approach the document information extraction task through Visual Question Answering, due to which Document Visual Question Answering is created nowadays. This domain is very young; therefore, most of the models and datasets are only in English. This paper proposes a plan to create a system that can improve non-English Document understanding, specifically for the Czech language. First, we are trying to form the first Czech dataset for Document VQA. Furthermore, we plan to develop the first model for DVQA, which will process Czech invoices. Last but not least, we propose to address three research questions extending the applicability of general document extraction technology to non-English languages.

2 Background

This section discusses the current situation in Document processing and Question answering. The first part focuses on a general overview of Intelligent Document Understanding and Visual Question Answering. Subsequently, we overview available datasets and models in these fields.

2.1 Intelligent Document Understanding

With Intelligent Document Understanding (IDU), machines are able to comprehend and analyze unstructured data. Earlier works for document understanding [8,12] stood on a predefined set of rules. Therefore they required an exact definition of a template for every type of document that they were processing. [17]

IDU combines Natural Language Processing (NLP), Computer Vision, Machine Learning, and Deep Learning. Transformers are best suited for these tasks; for example, simple transformers are practical for NLP and Computer Vision tasks. On the other hand, Layout-aware Transformers are used for IDU because they can comprehend the layout information for the given document. As a result, LayoutLM [19] combines text, document layout, and visual information to extract practical knowledge from a document.

2.2 Visual Question Answering

Question Answering (QA) [3] models work with text and retrieve answers to the given question [13] based on the information they got from the text. This process is a combination of natural language processing and information retrieval fields. On the other hand, Visual Question Answering (VQA) focuses on understanding the visual data. Even if the images contain some text, this text is not considered when answering the given questions. However, there is also a combination of QA and VQA, which incorporates text from the scene of the images.

Document Visual Question Answering [7] seeks to obtain knowledge from documents to answer questions. The asked questions may relate to different parts of the examined document, not only the text part; for example, they may refer to inserted images, tables, and forms, but they may also refer to the overall arrangement of the text. Therefore, for Document VQA, we also need to incorporate the detection of scene objects and an understanding of the document's layout and the relations between different parts of the layout.

Presently, there is lacking coverage of non-English models for Document Visual Question Answering. For this reason, we would like to expand the coverage of the models on Document VQA by the Czech language model.

Figure 1 presents an overview of our proposed system for document processing based on VQA. The model will be trained on our Czech dataset for Document VQA. The system will be able to answer the entered questions by writing the required answer and highlighting this answer in the document. Highlighting in the document is essential, mainly because the text of the correct answer may be in several places in the document, but the correct answer is only one of them. On the other hand, it is also possible that the correct answer will be found more than once in the text, and it is crucial to highlight all the right answers.

2.3 Datasets

This section describes datasets used for Document Question Answering or Named Entity Recognition.

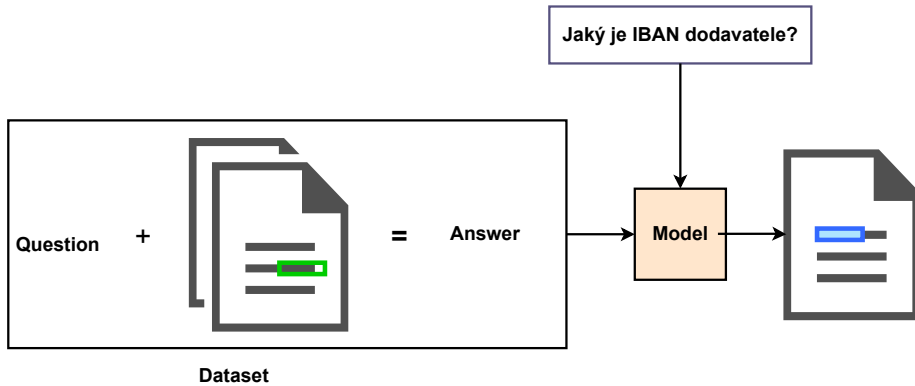


Fig. 1: Top-level approach for Document Visual Question Answering

Question Answering (QA) Nowadays, there are Question Answering datasets available in multiple languages. The most significant datasets are in English. These datasets differ in the size of question-answer pairs as well as the origin of these datasets. A large part uses texts from Wikipedia or newspaper articles, which are used to find answers to professional questions. Next, some datasets focus on technical areas, for example, on mathematical questions are also worth mentioning.

The first frequently used English dataset is the Stanford Question Answering Dataset (SQuAD) [13]. The original version of this dataset consists of over 100 000 questions posed on a set of more than 500 Wikipedia articles. The second version of the dataset has an additional 50 000 unanswerable questions. Conversational Question Answering (CoQA) [14] is another dataset in English. As is mentioned in the name of this dataset, it consists of more than 8 000 conversations and contains 127 000 questions, which also include evidence for answers. Answers can be free-form text.

The French Question Answering Dataset (FQuAD) [5] is an example of a dataset that is not in the English language. This french dataset is similar to the SQuAD dataset; both use Wikipedia articles. There are also two versions of FQuAD. The first one contains over 25,000 samples, and the second one is bigger, with more than 60,000 samples.

Although these datasets work with text, they cannot be used directly on Document VQA, given that they do not contain a visual stand. In Document VQA, it is crucial to look at the document from its visual standpoint; different parts of documents have different meanings. With the text-only QA datasets, this information is lost.

Visual Question Answering (VQA) In the VQA datasets, we can use the same images for different languages. This is possible because we focus on the objects in the picture, not the text it contains. One of the English datasets is VQA

dataset [1], which includes 204,721 images from the MS COCO dataset [10] with 614,163 questions and 7,984,119 answers. This dataset also possesses 50,000 abstract scenes with 150,000 questions and 1,950,000. The Image-Set Visual Question Answering (ISVQA) [2] dataset went one step further and focused on a multi-image setting. The dataset consists of 141,096 questions about objects and relationships in one or more images. In total ISVQA has 60,884 image sets.

The Indic Visual Question Answering dataset [4] is an example of a non-English VQA. The exciting thing about this dataset is that it focuses on three languages: Hindi, Kannada, and Tamil. This Indic dataset consists of 3.7 million image-question pairs for each of these three languages on the same set of images.

Named Entity Recognition (NER) Named Entity Recognition aims to locate and identify entities in the given text. One of the datasets for this task is the CoNLL-2003 dataset [15], which consists of two languages: English and German, and four types of named entities. The English data originate from Reuters Corpus, composed of 22,137 sentences. The German part consists of 18,933 from the ECI Multilingual Text Corpus, precisely from the German newspaper Frankfurter Rundschau. Another dataset for NER is Few-NERD [6]; this dataset is more extensive than CoNLL-2003 and consists of 188,238 sentences from Wikipedia, eight coarse-grained types, and 66 fine-grained types.

There are also datasets for legal documents, which are closer to our task. An example of one of these datasets, which is also non-English, is a Dataset of German Legal Documents for NER [9]. This dataset contains 66,723 sentences and two versions of annotations. The first version consists of 19 fine-grained semantic classes, and the second has seven coarse-grained classes.

Document Visual Question Answering datasets Only a few Document VQA datasets have been created recently, primarily in English. These datasets consist of web pages, scanned documents, or born-digital documents, and also various pages are from textbooks or posters.

Currently, the best dataset on Document VQA is DocVQA [11]. This dataset consists of several documents from the UCSF Industry Documents Library [18]. The important thing is that it also contains invoices, and all answers can be retrieved directly from the document's text. This is similar to the task of our future model; processing and analyzing invoices and developing an extractive model. As for the quality of the documents used, the dataset contains born-digital documents, scanned documents, and handwritten or typewritten documents. The reason why there are handwritten or typewritten documents is that the documents are from the period between 1960 and 2000. Overall the dataset consists of 50,000 questions over 12,767 document images, which are extracted from 6071 scanned documents. [11] Of the 50,000 questions, 36,170 of them are unique questions.

Visual Machine Reading Comprehension (VisualMRC) [16] is an example of a dataset consisting of screenshots of web pages. Another difference between this dataset and DocVQA is that it has images from different sources, which

increases its diversity and usability. Altogether the VisualMRC [16] consists of 30,562 questions, where 29,419 are unique.

2.4 Models

This section will discuss some applicable models for our Document Question Answering model and the models we are using for the baseline.

Question Answering models Question Answering models can retrieve the response to a question from a text. There are two different types of models based on their answers. The first type is Extractive QA; in this type, answers are directly written in the text. The second type is Generative QA, where the answer is a free text based on the context of the text. An example of a model for QA is the BERT Base model [3], which was fine-tuned on the SQuAD dataset.

LayoutLM family models The LayoutLM family consists of three generations of multimodal Transformer models, which were pre-trained on the IIT-CDIP Test Collection containing English scanned documents. Additionally, the LayoutLM family also offers one multilingual model trained in 53 languages, including Czech and Slovak.

3 Czech Document Visual Question Answering Dataset

This section will introduce our Document Visual Question Answering dataset in the Czech language. We will focus on collecting documents, creating questions, and mapping them to desired answers.

3.1 Data Collection

Our dataset contains 6,849 documents, the vast majority of which are invoices. The documents are primarily in the Czech and Slovak languages, but it also has Polish and Slovene languages. However, there are also non-Slavic languages like English, German, and Hungarian.

For each invoice, we asked questions about 15 entities it contains. For each entity, we created several different types of questions, covering multiple variants. This also helps the model to learn to answer more than one type of question for each entity.

In Figure 2, we can see how we created our dataset. At first, we have an invoice with several entities. We have chosen the 15 most important from them, for example, IBAN, account number, total sum to be paid, or invoice number. From annotators, we have obtained the exact positions of these entities on our invoices. Next, we created the questions for these entities, which were then mapped to the bounding boxes with the correct one or multiple answers.

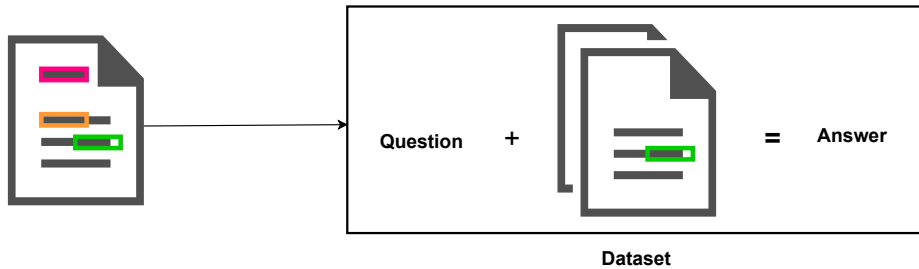


Fig. 2: Visual question answering dataset design: the green rectangle is the box that contains the information crucial to answer the question.

4 Research Proposal

This section introduces three research questions that we propose to study in our future work.

4.1 How can Textual QA and Visual QA datasets improve Document Understanding?

Our model will be trained on the Czech Document VQA dataset, which contains a large number of relevant, but *narrow* questions regarding the document's content. We hypothesize that if we first introduce our model to much more general and comprehensive QA datasets, the resulting model might be more robust in the unseen evaluation scenarios and hence, perform better as a result. Given that the textual QA datasets contain no layout, we propose to experiment with (i) a synthetic text layout and (ii) where applicable, the retrieval of the QA contexts from their original sources over the internet websites. Analogically, we would like to use the VQA dataset to see how it can improve the understanding of the document as an image. Finally, we would like to try all the relevant datasets and examine the results.

We will compare the performance of our Document VQA models to the baselines of (i) a competitive textual QA model, such as XLM-RoBERTa-Large, and (ii) the Visual Named Entity Recognition model, such as LayoutLM, trained solely on our Document VQA dataset. We will experiment with three models from the LayoutLM family: LayoutLMv2 Base, LayoutLMv2 Large, and Layout-XLM model.

4.2 How well can Document VQA models generalize beyond training entity types?

Conventional Named Entity Recognition models, for example, token classification, can identify only a closed set of entity types present in the training set and must be retrained when a new entity needs to be recognized. QA models can

theoretically circumvent this limitation by having the question as a part of the input. However, our model will be trained using a closed set of questions – one or a few for each entity type.

Therefore, it remains unclear whether such a QA model will be able to answer questions that are beyond the scope of its training set. We measure how well our model can generalize to unseen entities by selecting entity types as either training or evaluation and splitting the dataset accordingly.

To provide a reference to the results, we will compare our model with a competitive text-only model for QA and a model for the Visual Named Entity Recognition model from the LayoutLM family trained on the evaluation entities.

4.3 How much can Document VQA in non-English languages benefit from English datasets?

Lastly, we will quantify the benefit of utilizing English Document VQA datasets for document understanding in other languages. In this set of experiments, we will compare the model performance on a *target language*, i.e., a language of the final application, trained using (i) English data only, (ii) using a mixture of English and the target-language data, and (iii) using solely the target-language data. Details of the experimental setup can be found in Section 4.1.

Notably, the evaluation of the approach will assess the applicability of our models to *unseen languages*, which is necessary for relevancy in a vast majority of the world languages.

Our evaluation setup will include target languages where *any* document-understanding datasets are currently available. Thanks to our dataset, these will include Czech. Also, small-scale datasets for Document NER or QA are available in French and German.

5 Conclusion

This paper offers a basic overview of systems and datasets for Document Visual Question Answering. We created the first Czech dataset for Document VQA. Last but not least, we pose three research questions outlining future work: (i) improvement of Document Understanding, (ii) generalizing Document VQA beyond training entity types, (iii) benefit of English datasets for non-English Document VQA.


Acknowledgements We acknowledge the support of grant Intelligent Back Office, project number CZ.01.1.02/0.0/0.0/21_374/0026711.

References

1. Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C.L., Parikh, D.: VQA: Visual Question Answering. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2425–2433 (Dec 2015). <https://doi.org/10.1109/ICCV.2015.279>

2. Bansal, A., Zhang, Y., Chellappa, R.: Visual Question Answering on Image Sets. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) *Computer Vision – ECCV 2020*. pp. 51–67. Springer International Publishing, Cham (2020)
3. Chan, B., Möller, T., Pietsch, M., Soni, T.: Hugging face, <https://huggingface.co/deepset/roberta-base-squad2>
4. Chandrasekar, A., Shimpri, A., Naik, D.: Indic Visual Question Answering. In: 2022 IEEE International Conference on Signal Processing and Communications (SPCOM). pp. 1–5 (2022). <https://doi.org/10.1109/SPCOM55316.2022.9840835>
5. d’Hoffschmidt, M., Belblidia, W., Brendlé, T., Heinrich, Q., Vidal, M.: FQuAD: French Question Answering Dataset (2020). <https://doi.org/10.48550/ARXIV.2002.06071>
6. Ding, N., Xu, G., Chen, Y., Wang, X., Han, X., Xie, P., Zheng, H.T., Liu, Z.: Few-NERD: A Few-Shot Named Entity Recognition Dataset (2021). <https://doi.org/10.48550/ARXIV.2105.07464>
7. Ding, Y., Huang, Z., Wang, R., Zhang, Y., Chen, X., Ma, Y., Chung, H., Han, S.C.: V-Doc: Visual questions answers with Documents. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 21492–21498 (2022)
8. Klink, S., Dengel, A., Kieninger, T.: Document structure analysis based on layout and textual features. In: *Proc. of International Workshop on Document Analysis Systems, DAS 2000*. pp. 99–111 (2000)
9. Leitner, E., Rehm, G., Schneider, J.M.: A Dataset of German Legal Documents for Named Entity Recognition. *CoRR* **abs/2003.13016** (2020), <https://arxiv.org/abs/2003.13016>
10. Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft COCO: Common Objects in Context (2014). <https://doi.org/10.48550/ARXIV.1405.0312>, <https://arxiv.org/abs/1405.0312>
11. Mathew, M., Karatzas, D., Jawahar, C.: DocVQA: A Dataset for VQA on Document Images. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. pp. 2200–2209 (January 2021)
12. Niyogi, D., Srihari, S.N.: A Rule-Based System for Document Understanding. In: *Proceedings of AAAI-86*. pp. 789–793 (1986)
13. Rajpurkar, P., Jia, R., Liang, P.: Know What You Don’t Know: Unanswerable Questions for SQuAD (2018). <https://doi.org/10.48550/ARXIV.1806.03822>
14. Reddy, S., Chen, D., Manning, C.D.: CoQA: A Conversational Question Answering Challenge (2018). <https://doi.org/10.48550/ARXIV.1808.07042>
15. Sang, E.F.T.K., De Meulder, F.: Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. *CoRR* (2003). <https://doi.org/10.48550/ARXIV.CS/0306050>
16. Tanaka, R., Nishida, K., Yoshida, S.: VisualMRC: Machine Reading Comprehension on Document Images. *Proceedings of the AAAI Conference on Artificial Intelligence* **35(15)**, 13878–13888 (May 2021). <https://doi.org/10.1609/aaai.v35i15.17635>
17. Tito, R., Karatzas, D., Valveny, E.: Document collection visual question answering. In: *International Conference on Document Analysis and Recognition*. pp. 778–792. Springer (2021)
18. Industry documents library, <https://www.industrydocuments.ucsf.edu/>
19. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., Zhou, M.: LayoutLM: Pre-Training of Text and Layout for Document Image Understanding. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. pp. 1192–1200. KDD ’20, ACM, New York, NY, USA (2020). <https://doi.org/10.1145/3394486.3403172>

Manipulative Style Recognition of Czech News Texts using Stylometric Text Analysis

Radoslav Sabol and Aleš Horák 

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00, Brno, Czech republic
`{xsabol, haless}@fi.muni.cz`

Abstract. The rampant spread of manipulative texts purporting propaganda, disinformation or surveillance, requires the readers to take heed of the actual reasoning behind and the real purpose of the newspaper texts. The capability to recognize a malignant content asks for more and more concentration and background knowledge. A support offered by automated content analysis tools forms an important part of such protective approaches.

In the presented text, we introduce a new approach to detecting a set of manipulative stylistic techniques in Czech newspaper texts by exploiting stylometric methods in conjunction with deep learning text classification. We show that the stylometric analysis with almost 20,000 features allows to improve the results for most of the techniques. The results are evaluated with a previously published Czech Propaganda dataset.

Keywords: stylometry, propaganda detection, manipulative style analysis, Propaganda dataset, Czech

1 Introduction

The current accessibility and popularity of the Web, along with the ease of creating new content, takes freedom of expression to a whole new level, which is considered a positive development. An adverse side effect of this makes it straightforward to create websites and online news outlets that publish manipulative content. Disinformation through online news outlets creates an illusion of the information being reliable, affecting a much broader population than from the other sources [7]. Due to the immense and dynamic nature of the Internet, manual detection is difficult to grasp, and automated tools are desired to protect or warn readers of the manipulative content.

In 2019, a shared task was held in *Workshop on NLP4IF: censorship, disinformation, and propaganda*¹ [4]. The task consisted of two different problems based on the **Propaganda Techniques Corpus** [5] dataset. The dataset consists of fine-grained annotations that range from techniques that *leverage emotions* (for example *Loaded Language*, an act of using phrases with strong connotations) to *logical*

¹ <http://www.netcopia.net/nlp4if/2019/>

fallacies (like *Straw Man*, where writer refutes an argument not presented by the opposition). From 25 submitted approaches, the best-performing ones utilized BERT contextual embeddings. Other successful approaches exploited contextual embeddings of RoBERTa, ELMo, or context-independent representations based on lexical, sentiment, or TF-IDF features [4].

In the current paper, we present recent results in **manipulative style recognition** of Czech texts. In Section 2, we describe the specifics of the currently used benchmark dataset. Section 3 proposes a set of stylometric text features crafted using Czech linguistic tools. Section 4 presents a deep neural architecture based on XLM Roberta [3] that combines both the contextless stylometric features and the context-specific representation based on transformer models. In the last section, we evaluate and compare the approach that uses the proposed features and a similar model that does not.

2 Dataset Description

The *Propaganda* benchmark dataset, originally proposed by Baisa et al. [2], is a collection of 8,644 documents extracted from Czech news outlets that were previously investigated for spreading Russian propaganda [1]. The benchmark dataset is annotated with 21 diverse attributes, where 16 of them are relevant for analysis presented in this paper. The scope of annotation ranges from document-wide attributes to span level attributes that mark a specific segment of the text as an occurrence of a specific stylistic technique. The documents were tokenized and morphologically annotated using *unitok* [8], *majka* [11] and *desamb* [13] tools.

Among the annotated attributes, eight of them refer to *manipulative techniques*. The techniques are labeled at both the document and the span level.

- **Argumentation**: author presents an argument (yes/no)
- **Blaming**: author is blaming someone (yes/no)
- **Demonization**: author refers to individuals, groups, or political bodies as evil (yes/no)
- **Emotions** author uses emotive writing (fear, anger, indignation, compassion, other, missing)
- **Fabulation**: author spreads false rumors and exaggerates problems (yes/no)
- **Fear Mongering**: author appeals to fear, uncertainty, or certain threat (yes/no)
- **Labelling**: authors labels an entity with a short, pejorative phrase (yes/no)
- **Relativizing**: author either relativizes negative actions of Russia or positive actions of the opponent (yes/no)

Document-level attributes describe the expected structure and content of the document, so it is not reasonable to annotate them on the span level. These attributes are **Genre** (3 categories), **Topic** (13 categories), **Scope** (4 categories), **Location** (8 categories), and **Overall Sentiment** (3 categories).

Other attributes have annotations present on the span level, but they are not described as manipulative techniques. The listed attributes are **Expert** (yes/no), **Opinion** (yes/no), and **Russia** (5 categories).

3 Stylometric Text Features

In this section, we describe the proposed features that can be observed in Table 1. Overall, there is a lack of consensus for an ideal, universal set of stylometric features as they depend on the currently analyzed task and domain. The process of extracting such features requires linguistic analysis at various levels: *lexical, syntactic, semantic, structural, content-specific*, and *idiosyncratic* [6]. The current feature extraction implementation is inspired by the features proposed by [9].

Word and **Sentence length** distributions are implemented in three ways for both tokens and lemmas. The **naive** version is not adjusted to the real distribution present in the dataset and directly assumes word lengths 1 through 15 and more. **Improved** analysis creates bins of variable length derived from the data. The **N-Gram** approach analyzes naive word length n -grams.

Word Class N-Gram frequencies are extracted using annotations by *majka* and *desamb*. The N parameter ranges between 1 and 5, and only the n -grams with a relative frequency above 0.1% are considered. **Morphological Tags N-gram** frequencies consider more information than word classes. The **full** version uses the entire morphological tag, whereas the **simplified** omits the infrequent parts of the tag.

Word Repetition metrics are analyzed on multiple levels. **Average repetition** features compute frequency histogram for each unique token/lemma in the document, where the mean relative frequency is the resulting feature. **Bag of Words** repetition turns documents into TF-IDF normalized bag of words representation where stopwords and words with too low relative frequency are omitted. **Word Class Repetition** is a normalized word class histogram where for each token and its corresponding word class, the word class count is incremented for each sentence the token is repeated in. **Probabilistic Word Class Repetition** computes the probability of word class being repeated from the referential corpus and returns the difference between the referential probabilities and the probabilities from the provided document.

Letter Casing features are composed of two different methods. The first method computes n -grams of capital letters according to their position in the word and sentence using fixed rules. The indexed version also considers the exact position of the token in the sentence.

The **parametrized** version of **word suffixes** computes relative frequencies of the last n characters of each token document-wide. The **stemmed** version attempts to guess the word suffix based on the provided word and its corresponding lemma.

Word Richness considers two methods of computing vocabulary richness. *Simpson's Diversity Index* [10] is computed on all alphanumerical and alpha

tokens. *Coefficient of Colligation*, as known as *Yule's K* [12] is computed with the k values of 10, 100, 1,000 and 10,000.

Punctuation frequency examines the presence of various punctuation marks in the document. The **position frequency** version also considers the placement of punctuation marks. Finally, the n -gram version observes 100 most common punctuation n -grams with a relative frequency above 0.2%.

Fixed Typography is an idiosyncratic feature that checks for typography rules violations and various patterns related to typography that are checked using 11 regular expressions. **Dynamic Typography** observes the n -gram frequencies of non-alphanumeric tokens.

Character N-gram frequencies are extracted for at most 1,000 unique items with preferred document frequency around 50%. **Emoticon Presence** checks for the presence of pre-defined emojis in the presented document.

Table 1: Overview of proposed stylometric text features

Feature Type	Feature Subtype	# features	Language Independent
Word Length	naive	30	✓
	improved	77	✓
	n -grams	30	✓
Sentence Length	naive	25	✓
	improved	127	✓
	n -gram	25	✓
Word Repetition	avg. repetition per sent.	1	✓
	avg. repetition per doc.	1	✓
	word class repetition	13	
	prob. word class repetition	13	
	word repetition distance	12	✓
	bag of words repetition	100	✓
Word Class N-Grams	1 to 4-grams	514	
Morphological Tags N-Grams	full	10,000	
	simplified tags	200	
Letter Casing	1 to 3-grams	77	✓
	indexed 1 to 3-grams	417	✓
Word Suffixes	stemmed	100	✓
	parametrized n -grams	325	✓
Word Richness	richness metrics	6	✓
Stopwords	for lemmas	300	✓
	for tokens	300	✓
Punctuation	frequency	11	✓
	position frequency	60	✓
	n -gram frequency	76	✓
Typography	fixed rules	11	✓
	dynamic	100	✓
Character N-Gram Distribution	1 to 5-grams	6,550	✓
Emoticons Presence	n -grams	28	✓
Total		19,529	

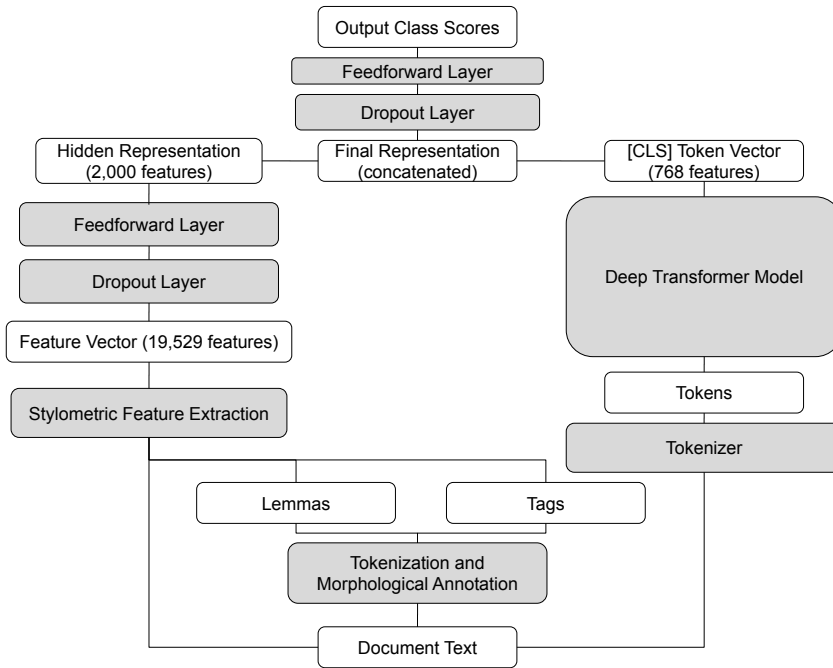


Fig. 1: Deep neural architecture for manipulative style detection

4 Detection Approach

The following section proposes an approach for manipulative style recognition using both stylometric text analysis and deep neural transformer models. A visual schema of this architecture is depicted in Figure 1.

4.1 Architecture Description

The input document is tokenized using `unitok` and morphologically annotated using `majka` and `desamb` tools. The resulting tokens, lemmas, and morphological tags are used to extract a feature vector using the stylometric analysis described in Section 3. The representation is then passed to a single feedforward layer. The resulting, more dense representation highlights the essential features for classification and represents the writing style used in the input document.

In tandem with the previous paragraph, the input text document is processed using **XLM Roberta Large** [3] deep transformer model that was pre-trained on 100 languages, the Czech language included. The model was pre-trained on sequences with a maximum token length of 512, so parts of the input document that exceed this limit are removed. The CLS token vector is extracted from the first item in the resulting sequence and it is then concatenated to the

hidden stylometric representation. In the final step, the concatenated representation is passed through the final feedforward layer, which models the predictions for each class in the attribute. The final prediction is selected using the *argmax* function.

4.2 Training Description

The proposed model is trained using the *HuggingFace* framework on 20 epochs. Hyperparameter values for the training were empirically determined. We use the learning rate of 3×10^{-6} and the linear warmup ratio of 0.1 for the AdamW optimizer. Due to the lack of training examples and label unbalance in some classes, more aggressive values for generalization were chosen. We use the dropout probability $p = 0.5$ for each presented feedforward layer, and a weight decay of 0.01 is used. The training computations were accelerated using GPU with a batch size of 32 and gradient accumulation to fit the GPU memory adequately.

5 Evaluation

In this section, we compare the proposed approach with two other approaches. The *dummy* baseline approach described in [1] always predicts the majority class. The second approach uses **XLM Roberta Large** with a standard HuggingFace classification head, where all the stylometric features are omitted.

The dataset is not split identically to Baisa et al. [1] because they use a different version of the *Propaganda* dataset. New train/test split was defined for the final version of the dataset, where 1,000 test examples are reserved for evaluation purposes. The testing set is identical throughout all evaluated attributes. Also, 500 examples were split as a development set for early stopping.

The experiments are performed for all the attributes mentioned in Section 2, where each setup is trained three times. The average **weighted F1** metric is computed from the three performed runs to factor in the label imbalance.

5.1 Results

Table 2 summarizes the performance of the proposed techniques for all attributes. The results showed that the dummy baseline was outperformed by a large margin in most categories. Notable exceptions can be seen in *Demonization* and *Relativization*, where the binary label imbalance is considerably higher than in other attributes.

The average weighted F1 score of the stylometric approach is lower than the text-only classification in cases of *Argumentation*, *Topic*, and *Labelling*. *Argumentation* is considered a complex and noisy attribute with a relatively low inter-annotator agreement. The current definition of *Argumentation* allows for anything from simple reasoning to a solid argument, along with some logical

Table 2: Summary of weighted F1 scores in % for the presented techniques. XLMR refers to XLM Roberta. Diff refers to the difference between the stylistic and non-stylistic XLM Roberta approach.

Attribute	Dummy	XLMR Large	XLMR Large with Stylometry	Diff
Argumentation	42.46	70.69	70.64	-0.05
Blaming	60.67	74.55	74.92	0.37
Demonization	95.67	96.13	96.19	0.06
Emotions	77.82	81.81	82.63	0.82
Fabulation	74.87	80.57	80.92	0.35
Fear Mongering	88.89	91.71	91.85	0.14
Labelling	76.7	83.37	83.09	-0.27
Relativizing	92.27	92.75	92.84	0.09
Genre	85.99	96.46	96.8	0.34
Topic	10.22	71.93	71.12	-0.81
Scope	41.03	89.36	90.15	0.79
Location	20.45	82.95	83.77	0.82
Sentiment	74.59	83.14	83.06	-0.08
Expert	39.03	76.1	77.42	1.32
Source	44.39	52.06	55.46	3.4
Opinion	80.52	87.61	88.35	0.74
Russia	53.12	82.88	83.63	0.75

fallacies, to be included in this class. Due to such high variation, the *Argumentation* may be challenging to grasp using automated machine learning methods. The difference between approaches is **0.05%** in favor of the non-stylistic one, which is considered a non-significant difference.

Topic results report **0.81%** difference in the favor of non-stylistic approach. The reason behind this difference may be that the attribute dwells in the semantics and the content of the document, but the proposed features specialize in non-content features. Thus in the learning process here, stylistic features may present overabundant information that degrades the overall performance.

The *Labelling* manipulative technique usually refers to a **small segment** of the text containing short, powerful phrase. Stylistic features, on the other hand, summarize the writing style of the **entire document**. The reason behind the **0.27%** decrease in the performance metric may be that the stylistic features could not properly capture this attribute's characteristics.

The most notable performance **increase** of **3.4%** can be seen with the *Source* attribute. The most important features responsible for the improvement relate to the presence and position of **capital letters**, as cited sources tend to be capitalized. Similar reasoning can apply to the *Expert* attribute, where a notable improvement is also present.

Another significant improvement of **0.82%** can be noticed with the *Emotions* attribute. The emotive writing style differs considerably from regular news, allowing for improved detection capabilities using the proposed features.

6 Conclusions and Future Directions

In the paper, we have evaluated a deep neural architecture that combines transformer models and stylometric text features to solve the manipulative style recognition task. We have introduced 13 feature categories from various levels of linguistic analysis, resulting in a feature vector of almost 20,000 dimensions. The results show that the proposed approach increases the performance for most Propaganda benchmark dataset attributes. The most notable increase was observed in the *Source* and *Expert* attributes, followed by the *Emotions* manipulative technique. It was also discovered that for some attributes (mainly *Topic*), the extracted non-content features do not relate to the specifics of the attribute, introducing noise and subsequently decreasing performance.

In the future development of the approach, we aim to increase the set of explored style attributes in both language independent and dependent features and evaluate the approach with other datasets and languages.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-CLARIAH-CZ project LM2018101. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme “Projects of Large Research, Development, and Innovations Infrastructures” (CESNET LM2015042), is greatly appreciated.

References

1. Baisa, V., Herman, O., Horak, A.: Benchmark dataset for propaganda detection in Czech newspaper texts. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019). pp. 77–83. INCOMA Ltd., Varna, Bulgaria (Sep 2019)
2. Baisa, V., Herman, O., Horak, A.: Manipulative propaganda techniques. In: RASLAN (2017)
3. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale. CoRR **abs/1911.02116** (2019), <http://arxiv.org/abs/1911.02116>
4. Da San Martino, G., Barron-Cedeno, A., Nakov, P.: Findings of the NLP4IF-2019 Shared Task on Fine-Grained Propaganda Detection. In: Proceedings of the second workshop on natural language processing for internet freedom: censorship, disinformation, and propaganda. pp. 162–170 (2019)
5. Da San Martino, G., Yu, S., Barrón-Cedeno, A., Petrov, R., Nakov, P.: Fine-grained analysis of propaganda in news article. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP). pp. 5636–5646 (2019)
6. Lagutina, K., Lagutina, N., Boychuk, E., Vorontsova, I., Shliakhtina, E., Belyaeva, O., Paramonov, I., Demidov, P.: A survey on stylometric text features. In: 2019 25th Conference of Open Innovations Association (FRUCT). pp. 184–195 (2019). <https://doi.org/10.23919/FRUCT48121.2019.8981504>

7. Martino, G.D.S., Cresci, S., Barrón-Cedeño, A., Yu, S., Di Pietro, R., Nakov, P.: A survey on computational propaganda detection. arXiv preprint arXiv:2007.08024 (2020)
8. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text tokenisation using unitok. In: Horák, A., Rychlý, P. (eds.) RASLAN 2014. pp. 71–75. Tribun EU, Brno, Czech Republic (2014)
9. Rygl, J.: Automatic adaptation of author's stylometric features to document types. In: International Conference on Text, Speech, and Dialogue. pp. 53–61. Springer (2014)
10. Simpson, E.H.: Measurement of diversity. *Nature* **163**(4148), 688–688 (1949)
11. Šmerk, P.: Fast morphological analysis of Czech. Proceedings of recent advances in slavonic natural language processing, RASLAN **2007**, 13–16 (2009)
12. Yule, C.U.: The statistical study of literary vocabulary. Cambridge University Press (2014)
13. Šmerk, P.: K počítačové morfologické analýze češtiny (Towards computer morphological analysis of Czech). Ph.D. thesis, Masaryk University, Faculty of Informatics, Brno (2010), <https://theses.cz/id/28r7vj/>

The Influence of a Machine Translation System on Sentiment Levels

Jaroslav Reichel  and Ľubomír Benko 

Constantine the Philosopher University in Nitra
94901 Nitra, Slovakia
{jreichel, lbenko}@ukf.sk

Abstract. The aim of the paper is to verify the influence of the used machine translation system on the level of sentiment in the translated text from Slovak to English using the available systems Google Translate and DeepL. The experiment was carried out on a parallel corpus created from subtitles of movies of different styles. The raw parallel corpus contained subtitles in Slovak and English. IBM Watson Natural Language Understanding service was used to identify the sentiment in the subtitles of ten movies of different genres. The paper also describes the methodology of preparing the dataset suitable for sentiment analysis using the IBM NLU service. The research showed a high correlation between human text and machine translation of subtitles for both translation systems. The research results show a high level of consistency of sentiment levels in both forms of translation. Based on the results obtained, the results of sentiment in machine translation can be generalized for the two most widely used translation systems.

Keywords: Machine translation, Natural language processing, Sentiment analysis, Slovak language

1 Introduction

The quality of machine translation depends on many factors. The text has many characteristics that need to be preserved in translation. However, there are also properties such as gender [1] or other regional formal habits that are not transferable between languages. A human translator can transfer some but machine translation has a problem with them.

An important characteristic of the text is the sentiment and the related emotion that the text should evoke. This is especially important in artistic texts such as poetry, prose, or film scripts. Emotion can also be captured in a text by observing an actor's performance. Sentiment and emotion can be identified in different ways. One of the most widely used is a tool from IBM that uses the IBM Watson supercomputer. For this purpose, the IBM Tone Analyzer tool was a service launched as an application programming interface (API) by IBM Corporation [2]. The IBM Watson™ Tone Analyzer service will be completely shut down in 2023 and is currently being replaced by the IBM Watson™

Natural Language Understanding service on IBM Cloud as part of IBM's service offerings. In the area of Natural Language Processing, researchers [3] compared DialogFlow, LUIS, and Watson, where Watson performed the best. IBM Watson Natural Language Understanding (IBM NLU) is a machine learning system that uses linguistic models to break free text into important words and phrases, called keywords. The program then calculates a general sentiment score for each keyword [4]. The *sentiment_score* variable indicates the sentiment measure, which takes values from -1 to 1.

Sentiment analysis is the field of studying and analyzing people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions [5]. Many researchers are investing a lot of energy in developing a sentiment analysis tool for different languages. This analysis has to take into account the creation of a large dictionary, the use of artificial intelligence, and so on. If it would be possible to use an already established tool, such as IBM Watson NLU service, for the Slovak language, these resources could be used more efficiently. The problem of using resource-rich languages [6,7] (typically English) for text identification in low-resource languages is dealt with in the area of cross-lingual sentiment analysis or classification [8].

The aim of this paper is to compare whether the results of sentiment analysis in machine translation differ between the two most widely used tools. For the analysis, it is necessary to obtain a parallel corpus of the Slovak language texts and the corresponding English human-written text. For this purpose, in this research movie subtitles will be used. However, this research will not use community-created subtitles but professional subtitles from a streaming service. These are high-quality human translations, which are also used, for example in English language teaching [9].

For the human translations, machine translations were obtained using the two most widely used online machine translation systems, Google Translate and DeepL. Their outputs were compared in the analysis. Using the IBM Watson™ Natural Language Understanding service were identified the sentiments of each segment in different versions of the translations: human-written text (EN), Google Translate machine translation from Slovak to English (GT), and DeepL machine translation from Slovak to English (DL).

The structure of the paper is as follows. The second section contains related work in the field of sentiment analysis. The third section describes the experimental setup, used dataset and applied research methodology. The subsequent section focuses on the research results based on the sentiment analysis and evaluation of the research problem. The fourth section offers a discussion of the results.

2 Related work

Sentiment analysis can be considered as a sub-field of information extraction [10]. Several commercial systems exist for sentiment analysis as Amazon Web Services Amazon Comprehend, Dandelion Sentiment Analysis API,

Google Cloud Platform Natural Language API, IBM Watson Natural Language Understanding, Lexalytics Semantria API, MeaningCloud Sentiment Analysis API, Microsoft Azure Text Analytics, ParallelDots Sentiment Analysis, Repustate Sentiment Analysis, Text2data Sentiment Analysis API, TheySay PreCeive API or twin word Sentiment Analysis API [11]. IBM Watson NLU is one such system, which provides sentiment analysis scores with great accuracy based on the information presented to it [12] and that was the reason it was used in the research. IBM Watson has been used by researchers in sentiment analysis often for different types of reviews [4] or social media posts [13].

Kapusta et al. [14] aimed to explore the influence of sentiment analysis on fake news identification. The most important finding was that there are statistically significant differences in the article sentiment where the fake news articles were identified with more negative sentiment. The authors used a basic sentiment classification method. Evaluating the assessment of the truthfulness of a text and its sentiment has also been addressed by Reichel et al. [15].

The scientific field that deals with sentiment analysis using multiple languages and machine translation is called Cross-lingual sentiment analysis (CLSA). CLSA leverages one or several source languages to help the low-resource languages perform sentiment analysis tasks. The models used in the CLSA methodology can be significantly refined if it is possible to find the best range of source languages for a given target language. The authors see the limitation mainly in the wrong models and data available for some languages, such as the Slovak language [16]. This area has been addressed by researchers for different languages or combinations of languages.

A comparative study [17] verifies sentiment classification from Chinese texts (reviews) using sentiment analysis in English. The authors compare sentiment from human translation and machine translation of publicly available services such as Google Translate, Yahoo Babel Fish, and Windows Live Translate.

3 Experimental setup

We addressed the following research question in the experiment:

RQ: Is there a significant difference between the translation systems Google Translate and DeepL in the accuracy of identifying sentiment scores compared to human texts?

We can infer the null hypothesis from it.

H0: There is no statistically significant difference between the correlation of sentiment level in human text and in Google Translate machine translation compared to the correlation of sentiment level in human text and in DeepL machine translation.

Through an experiment, we measure how much the sentiment score in the source text and the sentiment score in the machine translation from Google Translate match. In the same way, we will evaluate the level of agreement in sentiment between the original text and the machine translation from the DeepL system. We then compare these values. If the sentiment rates match across

translators, this will mean that the results of further analyses of sentiment in machine translation can be generalized to all common translation systems. A more detailed description of the research process is given in the following steps:

1. Data preparation
 - (a) Source corpus preparation
 - i. Alignment of Slovak and English subtitles into a coherent parallel corpus.
 - ii. Removal of erroneous, inconsistent, repetitive, or unnecessary records.
 - iii. Segmentation - Merging sentences that have been split to multiple subtitles back into a single segment.
 - (b) Generating a machine translation for each of the subtitles using Google Translate and DeepL machine translation systems.
 - (c) Identification of keywords and their sentiment using IBM Watson NLU service.
 - (d) Transforming the sentiment of the keywords into a coherent dataset of sentiment scores of each segment for the three sets:
 - i. Human text (EN),
 - ii. Machine translation from Google Translate (GT),
 - iii. Machine translation from DeepL (DL).
2. Data analysis
 - (a) Verification of the level of correlation of the identified sentiment of the machine translations (GT, DL) with the reference sentiment from the human text (EN).
 - (b) Comparison of results from Google Translate and DeepL.
3. Verification and interpretation of results
 - (a) Verification of research hypothesis H0.

Table 1: Sample of the dataset with subtitles and their machine translations from Google Translate and DeepL

id	Text_sk	Text_en	Text_gt	Text_dl
0	Blake.	Blake.	Blake.	Blake.
5	Zabalili nám jedlo?	Did they feed us?	Did they pack our food?	Did they pack us food?
6	Nie. Len poštu.	No. Just mail.	Not. Just mail.	No. Just mail.
7	Je čas na čaj!	Time for some tea!	It's tea time!	It's tea time!
8	Myrtle bude mať šteniatka.	Myrtle's having puppies.	Myrtle will have puppies.	Myrtle will have puppies.
10	Som hrozne hladný. Ty nie?	Oh, I'm bloody starving. Aren't you?	I'm terribly hungry. You do not?	I'm terribly hungry. Aren't you?

3.1 Machine translation generation

The corpus that was used contained 11 601 subtitles from 10 movies of different styles (war, fairy tale, action, sci-fi, comedy). The raw data had to be cleaned of erroneous entries, incorrectly paired parallel corpus pairs, and duplicate pairs. After resolving all errors, we obtained a parallel corpus of subtitles in English and Slovak. The created dataset contained 8551 segments.

To obtain the machine translation, the two most used online machine translation systems were chosen: Google Translate, and DeepL. Therefore, in the case of equality of results, it will be possible to generalize the results for machine translation obtained from different machine translation systems.

For further analysis, only the variables *id* (id of the segment), *Text_sk*, *Text_en*, and two variables *Text_gt* (machine translation obtained from Google Translate) and *Text_dl* (machine translation obtained from DeepL) were needed (Table 1).

3.2 Sentiment analysis

A tool IBM NLU was used for sentiment analysis. Each segment's sentiment analysis resulted in the identification of keywords and the determination of their sentiment. These results were transformed from JSON format into 3 matrices for each translation group (EN, GT, DL). There were 3 files with identified keywords for each segment and an associated *sentiment_score* value (Table 2). The output matrix from IBM NLU contains an identified *sentiment_score* for each keyword but for the same sentence (*id*) they match. Thus this is the *sentiment_score* of the sentence not of the keyword itself.

Table 2: Sample output from IBM NLU in the form of a matrix

id	keyword	text_en	sentiment_score
5	food	Did they feed us?	0
6	mail	No. Just mail.	0
7	tea time	Time for some tea! Tea's up!	0.842084
8	Myrtle	Myrtle's having puppies.	0.849348
8	puppies	Myrtle's having puppies.	0.849348
12	priesthood	It was the only reason I decided against the priesthood.	-0.839655

The results from the analysis using IBM NLU showed that there are approximately 18% fewer keywords in the machine translation compared to the human text (*Text_en* – 8419, *Text_gt* – 6886, *Text_dl* – 6923). After combining all three categories based on identified/unidentified sentiment, 4076 segments were extracted. These records were further manually cleaned of erroneous unpaired segments from sentences split into multiple subtitles. The resulting dataset contained 3768 segments.

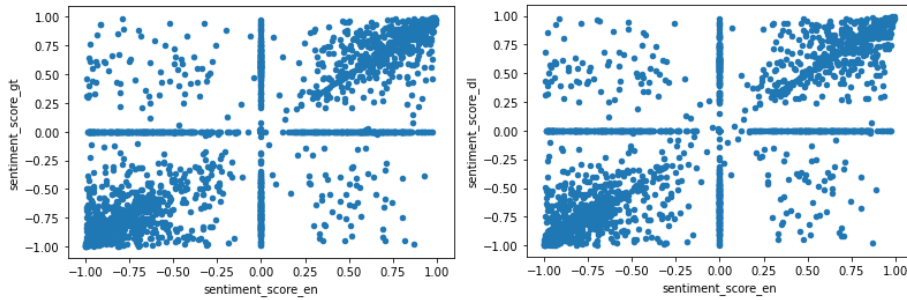


Fig. 1: 2D scatterplots for the correlation of the variable *sentiment_score_en* and a) *sentiment_score_gt*, b) *sentiment_score_dl*.

3.3 Results

RQ aims to verify whether there is a difference between Google Translate and DeepL in the results obtained. H_0 predicts that there is no statistically significant difference between the correlation of sentiment level in human text and in Google Translate machine translation compared to the correlation of sentiment level in human text and in DeepL machine translation. Correlation analysis was used to verify the dependence. Correlation analysis verifies, in a simplistic way, that if sentiment is high in human text, it is also high in machine translation and vice versa. To determine the correct method for correlation analysis, the distribution of each group of EN, GT and DL was verified. The *sentiment_score* variable does not have a normal distribution. This is confirmed by the results of the Kolmogorov-Smirnov test for all 3 variables: *sentiment_score_en* ($D(3768) = 0.256, p < 0.01$), *sentiment_score_gt* ($D(3768) = 0.276, p < 0.01$) and *sentiment_score_dl* ($D(3768) = 0.272, p < 0.01$). Since enough cases are available, the parametric method can be used: Pearson's correlation coefficient was used. The calculation was performed at a 5% significance level.

Results of correlation analysis (Fig. 1):

- EN/GT: $r(3768) = 0.73, p < 0.01$,
- EN/DL: $r(3768) = 0.74, p < 0.01$.

From the graph (Fig. 1), quite a large number of pairs contain the value 0 in at least one of the variables. This means that sentiment has not been identified in any of the translations (of the pair). If we exclude these segments from the analysis in order to mainly evaluate the match in identified sentiment, then the results look like the following (Fig. 2):

- EN/GT: $r(1497) = 0.86, p < 0.01$,
- EN/DL: $r(1539) = 0.86, p < 0.01$.

Considering the correlation results between the human text and the machine translations (0.73 and 0.74 in the unadjusted dataset (Fig. 1); 0.86 and 0.86 in the

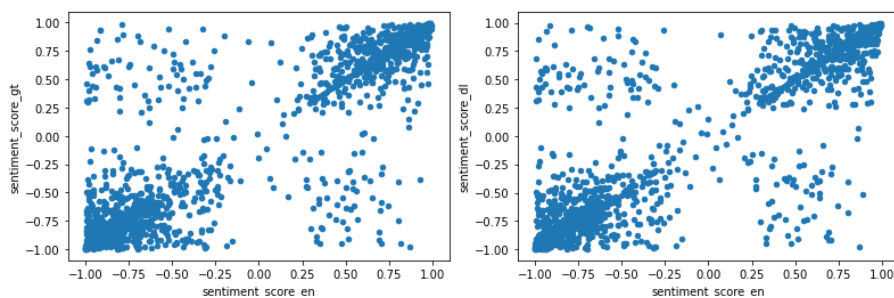


Fig. 2: 2D scatterplots for the correlation of the variable *sentiment_score_en* and a) *sentiment_score_gt*, b) *sentiment_score_dl* except for segments with neutral sentiment.

adjusted dataset (Fig. 2)), it can be argued that there is no significant difference between them. H_0 is thus not rejected. Hence, the results of Google Translate are significantly similar to the results of DeepL and therefore it is relevant to use only one of these systems in further analysis. Based on the rejection of H_0 , these results can be generalized.

4 Conclusion

We tested whether there is a significant difference in sentiment analysis for texts translated by Google Translate and DeepL. The results say that there is no difference. This means that for further sentiment analysis in machine translation, it is not necessary to do duplicate analyses for multiple translation systems but just choose which one suits better based on text style. The results of the research can be generalized for machine translation from Slovak to English.

By evaluating RQ, it was verified that there is no significant difference in sentiment transfer in machine translation between the most widely used machine translation systems, i.e. Google Translate and DeepL. It is therefore possible to generalize the results for machine translations in general.

Acknowledgement This work was supported by the Slovak Research and Development Agency under the contract No. APVV-18-0473 and by the projects UGA VII/20/2022 and UGA VII/1/2022.

References

1. Rabinovich, E., Mirkin, S., Patel, R. N., Specia, L., Wintner, S.: Personalized machine translation: Preserving original author traits In: 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference, vol. 2, pp. 1074–1084 (2017), doi: 10.18653/v1/e17-1101.

2. Al Marouf, A., Hossain, R., Kabir Rasel Sarker, Md. R., Pandey, B., Tanvir Siddiquee, S. Md.: Recognizing Language and Emotional Tone from Music Lyrics using IBM Watson Tone Analyzer. In: 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Feb. 2019, pp. 1–6. doi: 10.1109/ICECCT.2019.8869008.
3. Canonico, M., de Russis, L.: A comparison and critique of natural language understanding tools. In: https://thinkmind.org/download.php?articleid=cloud_computing_2018_6_20_20057, last accessed 2022/06/23.
4. Lu, T. J., Nguyen, A. X. L., Trinh, X. V., Wu, A. Y.: Sentiment Analysis Surrounding Blepharoplasty in Online Health Forums. In: *Plastic and Reconstructive Surgery - Global Open*, vol. 10, no. 3, Mar. 2022, doi: 10.1097/GOX.0000000000004213.
5. Liu, B.: Sentiment analysis and opinion mining. In: *Synthesis Lectures on Human Language Technologies*, vol. 5, no. 1, pp. 1–184, 2012, doi: 10.2200/S00416ED1V01Y201204HLT016.
6. Ghafoor, A. et al.: The Impact of Translating Resource-Rich Datasets to Low-Resource Languages through Multi-Lingual Text Processing. In: *IEEE Access*, vol. 9, pp. 124478–124490, 2021, doi: 10.1109/ACCESS.2021.3110285.
7. Baisa, V.: Czech Grammar Agreement Dataset for Evaluation of Language Models. In: Horak, A., Rychly, P., Rambousek, A. (eds.) *RASLAN 2016*, pp. 63–67. Karlova Studanka, Czech republic (2016).
8. Zhou, X., Wan, X., Xiao, J.: Cross-Lingual Sentiment Classification with Bilingual Document Representation Learning. In: *Association for Computational Linguistics*, pp. 1403–1412 (2016)
9. Dizon, G., Learning, B.: Language learning with Netflix: Exploring the effects of dual subtitles on vocabulary learning and listening comprehension. In: *Computer Assisted Language Learning Electronic Journal*. no. 3, pp. 52–65 (2021)
10. Sazzed, S., Jayarathna, S.: A sentiment classification in bengali and machine translated english corpus. In: *Proceedings - 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science, IRI 2019*, pp. 107–114 (2019) doi: 10.1109/IRI.2019.00029.
11. Ermakova, T., Henke, M., Fabian, B.: Commercial Sentiment Analysis Solutions: A Comparative Study. In: *17th International Conference on Web Information Systems and Technologies*. pp. 103–114. (2021) doi: 10.5220/0010709400003058.
12. Dash, A. S., Pathare, S. B.: Survey of Sentiment Analysis Through Machine Learning for Forecasting Indian Stock Market Trend. In: *SSRN Electronic Journal*. (2022) doi: 10.2139/SSRN.4057274.
13. Daneshfar, Z., Asokan-Ajitha, A., Sharma, P., Malik, A.: Work-from-home (WFH) during COVID-19 pandemic – A netnographic investigation using Twitter data. In: *Information Technology & People*, (2022), doi: 10.1108/ITP-01-2021-0020.
14. Kapusta, J., Benko, L., Munk, M.: Fake News Identification Based on Sentiment and Frequency Analysis. pp. 400–409, 2020, doi: 10.1007/978-3-030-36778-7_44.
15. Reichel, J., Magdin, M., Benko, L., Koprda, Š.: Can Fake News Evoke a Positive/Negative Affect (Emotion)? In: *DIVAI 2020*, pp. 563–572. (2020)
16. Xu, Y., Cao, H., Du, W., Wang, W.: A Survey of Cross-lingual Sentiment Analysis: Methodologies, Models and Evaluations. In: *Data Science and Engineering*, vol. 1, p. 3, (2022), doi: 10.1007/s41019-022-00187-3.
17. Wan, X.: A comparative study of cross-lingual sentiment classification. In: *Proceedings - 2012 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2012*, pp. 24–31. (2012). doi: 10.1109/WI-IAT.2012.54.

Subject Index

- AHISTO 157
- BLEU 71
- Bloom filter 91
- classification 15
- corpus 65, 91, 105, 113, 141
- corpus statistics 173
- corpus tools 123
- Czech 79, 163, 191
- dataset split 131
- deduplication 91
- dialogue dataset 131
- dictionary editor 55
- document visual question answering 183
- EHR 163
- embedding 15
- Estonian 3
- evaluation 47, 71, 79, 97
- FastText 79
- genre 141
- health records 163
- information extraction 35
- inter-annotator agreement 97
- Italian 105
- keyword extraction 25, 47
- language pipeline 123
- Lexonomy 55
- Lombard 105
- machine translation 47, 71, 201
- manipulative style analysis 191
- meaning 65
- medieval texts 157
- model compression 79
- multi-modal learning 35
- multiword prepositions 173
- NVH 55
- OCR 35, 157
- online conversation 131
- parallel corpus 105
- parallel data 3
- parsing 173
- Persian 113
- Polish 163
- POS tagging 97
- prepositions 173
- propaganda 191
- question answering 183
- Russian 25, 173
- sentiment analysis 201
- Sketch Engine 123
- Slovak 3, 71, 163, 201
- structured documents 35
- style analysis 191
- stylometry 191
- tagger 79
- tagging evaluation 97
- text annotation 141
- tokenizer 149
- transformers 35
- translation extraction 3
- verb 65
- visual question answering 183
- web corpus 141
- web-crawled corpus 113
- XML 55

Author Index

- Anetta, K. 163
- Bankovič, M. 35
- Benko, V. 113
- Benko, L. 71, 201
- Benkova, L. 71
- Boyarsky, K. 173
- Denisová, M. 3
- Forgáč, F. 47
- Geletka, M. 35, 183
- Herman, O. 91
- Horák, A. 157, 191
- Jakubíček, M. 55
- Kadlčík, M. 183
- Kanevsky, E. 173
- Kelebercová, L. 47
- Khokhlova, M. 25
- Koryshev, M. 25
- Kostka, M. 123
- Kovář, V. 55
- Kozlova, A. 173
- Kraus, J. 141
- Lebedíková, M. 131
- Meluš, D. 35
- Mikušek, O. 15
- Měchura, M. 55
- Nevěřilová, Z. 79
- Novotný, V. 157
- Ohlídalová, V. 97
- Plhák, J. 131
- Rambousek, A. 55
- Reichel, J. 201
- Rychlý, P. 149
- Sabol, R. 191
- Ščavnická, Š. 35, 183
- Signoroni, E. 105
- Šmahel, M. 131
- Sojka, P. 35, 183
- Sotolář, O. 131
- Špalek, S. 149
- Stará, M. 65
- Štefánik, M. 35, 183
- Suchomel, V. 141
- Tkaczyk, M. 131
- Zakharov, V. 173

RASLAN 2022

Sixteenth Workshop on Recent Advances in Slavonic Natural Language Processing

Editors: Aleš Horák, Pavel Rychlý, Adam Rambousek

Typesetting: Adam Rambousek

Cover design: Petr Sojka

Published and printed by Tribun EU

Cejl 892/32, 602 00 Brno, Czech Republic

First edition at Tribun EU

Brno 2022

ISBN 978-80-263-1752-4

ISSN 2336-4289