RASLAN 2019
Recent Advances in Slavonic Natural Language Processing

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

RASLAN 2019

Recent Advances in Slavonic Natural Language Processing

Thirteenth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2019
Karlova Studánka, Czech Republic, December 6–8, 2019
Proceedings

Tribun EU
2019

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

ISBN 978-80-263-1530-8
ISSN 2336-4289

Preface

This volume contains the Proceedings of the Thirteenth Workshop on Recent Advances in Slavonic Natural Language Processing (RASLAN 2019) held on December 6th–8th 2019 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 2019 offers a rich program of presentations, short talks, technical papers and mainly discussions. A total of 17 papers were accepted, contributed altogether by 28 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 Marie Stará. The TEXpertise of Adam Rambousek (based on LATEX 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 2019 Karel Pala

Table of Contents

IV Text Corpora

Part I

Morphology and Syntax

Comparing majka and MorphoDiTa for Automatic Grammar Checking

Jakub Machura, Helena Geržová,
Markéta Masopustová, and Marie Valíčková

Faculty of Arts, Masaryk University
Arne Nováka 1, 602 00 Brno, Czech Republic
{415795,400133,428801,415295}@mail.muni.cz

Abstract. Developing a grammar checker requires the most accurate morphological analysis. We have been using the majka analyzer and DESAMB tagger so far, but due to certain obstacles to disambiguation, we encountered many errors in morphological analysis. Nowadays, there are several tools that achieve comparable results. Therefore, it was beneficial to test the one which is well-kept and open-source – the MorphoDiTa system. For the detection of grammatical, stylistic and punctuation errors we use mainly special grammars built into the SET parser and this paper presents results based on outputs of both morphological analyzers.

Keywords: syntactic analysis, SET, grammar checker, punctuation, comma, homonymy, grammatical agreement, subject-predicate agreement, colloquial expressions, zeugma

1 Introduction

To write a text without any grammatical, spelling, or typographical mistake[1] is one of the main features of high-standard typed text. Nowadays, users of language more often create demand for having software which would reliably detect and correct various kinds of mistakes in texts.

In the Czech environment are known two commercial grammar checkers: 1. Grammar checker built into the Microsoft Office, developed by the Institute of the Czech language [14] and 2. Grammaticon checker made by the Lingea company [1]. From February 2019 the Masaryk University in collaboration with the Charles University, the Institute of the Czech language, and the Seznam company has started a new project of developing an automatic online language checker [5].

This paper is aimed at a description of some well-know as well as particular obstacles in the morphological analysis. The paper also contains an important result: comparison and evaluation of two systems for the morphological analysis.

The structure of the paper is in the following way: The next section superficially describes several tools used for automatic text analyses and

[1] In this paper we use terms error and mistake synonymously as equivalents of *chyba* in Czech.

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 3–14, 2019.

thoroughly examines two examples of phenomena which cause us obstacles during the morphological analysis. Then follows the comparison and evaluation of two examined systems.

2 Some Components For Automatic Language Checking

2.1 The SET System

For the main purposes of the new project is used the SET parser developed by Kovář [10]. In order to detect more complicated grammatical mistakes automatically (e. g. the subject-predicate agreement, punctuation errors, ...), any grammar checker should work with an output of the morphological analysis which means that for every single word in the sentence structure must be assigned lemma and morphological tag. Nowadays, two mainly used conceptions exist on how to represent grammar information of Czech words – attributive, and positional tag system. An advantage of the SET parser is not only an ability to work upon with the attributive as well as the positional tag sets, but the SET system (which was primarily designed as the syntactic parser [8]) contains functionalities which deal with partial grammar checking.

2.2 Tools Used For the Morphological Analysis and Their Obstacles

Up to now, we have used `unitok` tokenizer [11] and for the morphological analysis, the analyzer majka [17] and subsequent disambiguation is operated by the DESAMB [16] tagger. The analysis using both tools brings sufficient results, yet some inaccuracy has occurred. Of course, there are factors which prevent absolute accuracy for the automatic morphological analysis, such as homonymy of word forms, especially for flective languages, or grammar mistakes caused by users of language:

Ex. 1
S1: Muž, který *je*$_1$ pravděpodobně unesl, *je*$_2$ běloch.
(*The man, who probably kidnapped them*$_1$, *is*$_2$ *a white man.*)

The analyzer majka and DESAMB tagger give following output for S1 (1^{st} column word, 2^{nd} column lemma and 3^{rd} column tag):

```
<s>
Muž                 muž                 k1gMnSc1
<g>
,                   ,                   kIx,
který               který               k3yRgMnSc1
je1                 on                  k3xPp3gNnSc4
pravděpodobně       pravděpodobně       k6eAd1
unesl               unést               k5eAaPmAgMnS
<g/>
,                   ,                   kIx,
je2                 být                 k5eAaImIp3nS
běloch              běloch              k1gMnSc1
<g/>
.                   .                   kIx.
</s>
```

The word `je` in S1 are tagged as a pronoun (`k3xPp3gNnSc4`), and as a form of the verb to be (`k5eAaImIp3nS`).

Nevertheless, many users of Czech, even natives speakers, often forget putting the second comma which set an inserted subordinate clause apart from a main clause from the right side:

S2: *Muž, který je_1 pravděpodobně unesl je_2 běloch.*

The analyzer majka and DESAMB tagger now provide this output for S2:

```
<s>
Muž                 muž                 k1gMnSc1
<g/>
,                   ,                   kIx,
který               který               k3yRgMnSc1
je1                 on                  k3xPp3gNnSc4
pravděpodobně       pravděpodobně       k6eAd1
unesl               unést               k5eAaPmAgMnS
<g/>
je2                 on                  k3xPp3gNnSc4
běloch              běloch              k1gMnSc1
<g/>
.                                       kIx.
</s>
```

The absence of comma from the right side of the subordinate clause caused the tagger to choose a wrong (but justifiable) tag for the je_2 in the S2.

Ex.2

To deal with the case homonymy within a paradigm of one noun could represent even much more difficult task for any tagger compared with word

form homonymy. For instance, the word form *koření* is homonymous for six cases of the singular and four cases of the plural form of noun *koření* (spice) (See the Table 1), and at the same time could be a form of the verb *kořenit* (spice up) (See the Table 2). Additionally, the instrumental case of the singular form and the accusative case of the plural form are also homonymous (*kořením*). Homonymous forms in Tables 1 and 2 have highlighted background.

Table 1: Paradigm of the noun *koření* (spice).

Paradigm of *koření*		
	singular	plural
nominative	koření	koření
genitive	koření	koření
dative	koření	kořením
accusative	koření	koření
vocative	koření	koření
locative	koření	kořeních
instrumental	kořením	kořeními

Table 2: Conjugation of the verb *kořenit* (Spice Up) – Present Tense.

Conjugation of *kořenit*		
	singular	plural
1^{st} person	kořením	kořeníme
2^{nd} person	kořeníš	kořeníte
3^{rd} person	koření	koření

S3: V druhém šuplíku najdeš správné **koření**.
(*In the second drawer, you will find the right spice.*)

The analyzer majka and DESAMB tagger provide the output for S3:

```
<s>
V                    v                    k7c6
druhém               druhý                k4xOgInSc6
šuplíku              šuplík               k1gInSc6wH
najdeš               najít                k5eAaPmIp2nS
správné              správný              k2eAgNnSc1d1
koření               koření               k1gNnSc1
<g/>
.                    .                    kIx.
</s>
```

In the S3, the DESAMB tagger wrongly identified the case as the nominative for the noun *koření* (the verb *najít* requires an object in the accusative case though). Moreover, the finite verb *najdeš* according to the ending *-š* contains information that a subject is a 2^{nd} person of the singular and the subject *ty* (you) is visibly unexpressed. Therefore, no other constituent in the clause should have a form in the nominative case.

S4: Běž a rychle kup **koření** v supermarketu.
(*Run and buy quickly spice/spices in the supermarket.*)

The analyzer majka and DESAMB tagger provide the output for S4:

```
<s>
Běž                 běžet               k5eAaImRp2nS
a                   a                   k8xC
rychle              rychle              k6eAd1
kup                 kup                 k1gInSc1
koření              kořenit             k5eAaImIp3nS
v                   v                   k7c6
supermarketu        supermarket         k1gInSc6
<g/>
.                   .                   kIx.
</s>
```

In the S4, the noun *koření* was incorrectly tagged as a verb instead of a noun in the accusative case and the imperative form *kup* of the verb *koupit* got the tag as a noun in the nominative case.

We also noticed that the DESAMB tagger sometimes matches some adjectives and pronouns as nouns (See Ex. 3). Nevertheless, they stand in the position of premodifier followed by a noun. Thus, there could not be any syntactic reason to tag them as nouns.

Ex.3

Do své hospody vezmu jen slušné zákazníky, ne **žádné** [k1gMnPc4] vagabundy.
(*I will take only polite customers to my pub, not any vagrants.*)

Žádný `[k1gMnSc1]` kout světa není bezpečný.
(*There is no place in the world that is safe.*)

S **dalším** `[k1gNnSc7]` naším vlastníkem uzavřeli dohodu.
(*They made an agreement with our other proprietor.*)

Dokázal uhrát **stejnou** `[k1gFnSc4]` plichtu i s Francií.
(*It managed to play a tied score also with France.*)

2.3 Making Use of the MorphoDiTa system

Collaboration with linguists from the Charles University and inaccuracies described above led us to start thinking about a possibility to use the MorphoDiTa[2] – a complex tool for the morphological analysis which is used especially at The Institute of the Czech National Corpus and The Institute of Theoretical and Computational Linguistics of Faculty of Arts, Charles University. The open-source MorphoDita [18] is an acronym from the Morphological Dictionary and Tagger. It uses an accessible and updated morphological dictionary MorfFlexCZ [4] and it is composed of several modules (a tokenizer, Morphological Generation, Morphological Analysis and Morphological Tagger [15]). Additionally, the MorphoDiTa system achieves one of the best results in accuracy to assign a tag in comparison to other Czech systems [16].

2.4 Positional - attributive tags conversion

The MorphoDiTa system works with positional tag set. As was mentioned earlier, the SET parser also has functionality which allows processing morphological tags in positional format. The `--posttags` switch provides conversion of positional tags into the attributive format (See the process of conversion below).

The output of the morphological analysis provided by the MorphoDiTa system and follow-up conversion:

S1: *Muž, který je pravděpodobně unesl, je běloch.*

```
                                                --posttags
Muž             muž             NNMS1-----A----  k1gMnSc1eA;cap
,               ,               Z:-------------  kI
který           který           P4YS1----------  k3yRgMgInSc1
je              on              PPXP4--3-------  k3xPg.nPc4p3
pravděpodobně   pravděpodobně   Dg-------1A----  k6d1eA
unesl           unést           VpYS---XR-AA---  k5mAgMgInSp.mReA
,               ,               Z:-------------  kI
je              být             VB-S---3P-AA---  k5mInSp3mIeA
běloch          běloch          NNMS1-----A----  k1gMnSc1eA
.               .               Z:-------------  kI
```

It should be noted that the conversion is not going on in the ration 1:1 which means that not every single tag from the positional tag set has a corresponding tag in the attributive tag set.

[2] The open-source version is available on `https://lindat.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0023-43CD-0`.

3 Partial Grammar Checking Using the MorhoDiTa

3.1 Punctuation Checking

Automatic punctuation checking (finding a place where a missing comma should be inserted, or removal of an incorrectly written comma) belongs to a much more complicated grammar task. The SET system has functionality which reaches one of the best results in finding a place where a missing comma should be inserted [9]. Testing and comparison (majka + DESAMB versus MorphoDiTa) were made on the DESAM corpus [13] that contains 61 098 commas. For the purpose of the task to find a place where a missing comma should be inserted, every single comma was removed from the corpus and the SET system works with a plain text without any comma. Generally, the MorhoDiTa did not bring better results than the majka and the DESAMB tagger. We assume, though, the MorphoDiTa deals with case homonymy a bit better, but this assumption needs deeper research (See the Table 3).

Table 3: Results of the comparison – Punctuation checking. TP – True Positives (correctly found commas); FP – False Positives (incorrectly found commas); FN – False Negatives (missed commas), P – Precision; R – Recall. 1. Rules which deal with commas after the connector; 2. Rules for multiple sentence members mostly based on case agreement; 3. Rules for multiple sentence members mostly based on case agreement + information about collocation from the corpus csTenTen17 [19].If we consider whole rules as complex determining the insertion of commas, the majka and the DESAMB tagger win both in precision and recall. However, it is worth noticing that the MorphoDiTa has better precision in the detection of groups of nouns in coordinating relation (which share a case form), but with a lower value of recall.

Total of commas: 61 098	**majka + DESAMB**					**MorhoDiTa**				
Rules	**TP**	**FP**	**FN**	**P (%)**	**R (%)**	**TP**	**FP**	**FN**	**P (%)**	**R (%)**
All rules	33 833	2 457	27 265	**93,23**	**55,37**	33 808	2 741	27 290	92,50	55,33
1. Connector	32 806	2 256	28 292	**93,57**	53,69	32 805	2 609	28 293	92,63	53,69
2. Coordination	1 025	224	60 073	82,07	**1,68**	1 005	145	60 093	**87,39**	1,64
3. Coordination	1 034	94	60 064	91,67	**1,69**	804	56	60 294	**93,49**	1,32

3.2 Automatic detection of zeugma

The diploma thesis of Geržová [3] deals with automatic detection of zeugma. In Czech, this term means that one expression is in semantic or syntactic relation with two other paratactically connected expressions (e.g. two verbs), but the whole structure is grammatically defective [7] as in the example below.

Ex.:
Potvrzujete a souhlasíte s tím, že žádný software není bez vad.
(*We confirm and agree with an idea that no software is without any fault*.)

In the thesis [3], Geržová focused mainly on verbal coordinations. For this purpose we created eighty-three rules based on the assumption that the first verb with the obligatory subject in coordination with the second verb does not have a suitable addition in the sentence. Together the rules had precision 63,36 % and recall 54,78 % [3].

However, many of the false positives were caused by inaccurate disambiguation, especially if there was ambiguity in-between cases (as in the example below).

Ex.:
Řekl bych, že **věc** `/k1gFnSc4/` **chápe a rozumí grafům**.
(*I would say he gets the point and understands graphs*.)

Therefore we supposed that a more accurate morphological analyzer could increase precision and recall. The precision of the rules in the above-cited work was measured on the first 100 million lines of the corpus csTenTen17 [19]. Recall was obtained from another file ("test_set_with_errors_2") with errors of this type.

To compare majka (DESAMB) and MorphoDiTa, we chose twenty verbs and adequate rules for the detection of zeugma. Against the diploma thesis [3], we tested precision for this time on a file ("test_set_mixed_1") that contained one thousand sentences for each tested verb (rule). These sentences included coordinating structures consisting of a tested verb and another verb. With this method we obtained more defect structures of this type, because as we learned in theses [3] zeugma is not a very frequent phenomenon. The results are included in Table 4.

Table 4: TP test_set_mixed_1 – true positives found in file "test_set_mixed_1"; FP test_set_mixed_1 – false positives incorrectly marked as zeugma in file "test_set_mixed_1"; Precision test_set_mixed_1 – based on results from file "test_set_mixed_1"; TP test_set_with_errors_2 – true positives found in file "test_set_with_errors_2"; FN + TP – number of all zeugmas in the tested file "test_set_with_errors_2".

	test_set_mixed_1			test_set_with_errors_2		
	TP	FP	Precision (%)	TP	FN + TP	Recall (%)
majka + DESAMB	314	57	84,64	227	483	47,00
MorphoDiTa	359	50	87,78	225	483	46,58

According to the results of the analysis, values of precision and recall were similar for both tested analyzers. The rules using the MorphoDiTa morphological analyzer had a few percent higher precision and a half percent worse recall.

3.3 Automatic detection of errors in subject-predicate agreement

Another part of comparison and evaluation was made on sentences that contain errors of subject-predicate agreement. Results of the majka analyzer on this kind of data were discussed in [12].

During the testing, we looked at the subject-predicate agreement with a simple subject that was expressed within a given clause. We realised the complexity of the task, therefore, we decided from the testing to exclude examples where a subject is expressed elsewhere within the sentence or is not expressed at all. At the same time, we removed phrases contained errors that are not covered by the existing rules.

We used a file with 124 sentences, of which 34 were correct and 90 contained one or more errors of subject-predicate agreement. The table of results (majka, MorphoDiTa) is attached to the end of this subchapter (Table 5). The first part of the testing revealed that majka + DESAMB behave cautiously – rather avoid to make a false report about the incorrect subject-predicate agreement, but at the same time, lots of real mistakes are ignored. On the other hand, the MorphoDiTa reports more mistakes, even though lots of them are evaluated as false reports.

In the case of majka and DESAMB, the majority of the false reports are caused by the wrong disambiguation – the DESAMB tagger often identifies a subject as a noun in accusative form and then the SET parser assesses the noun as an object [5].

Table 5: Comparison of majka and MorphoDiTa on the identification of subject-predicate agreement. TP – revealed error when SET correctly labeled a subject as predictive on predicate and labeled it "subject-bad"; FP – so called fake mistake where a "subject-bad" tag was wrongly labbeled to another member in the clause. FN – missed errors in subject-predicate agreement.

	TP	FP	FN	Precision (%)	Recall (%)
majka	29	15	65	65,9	30,9
MorphoDiTa	40	48	54	45,5	42,6
MorphoDiTa (after repair)	40	12	54	76,9	42,6

A deeper inspection of MorphoDiTa's results revealed that the positional-attributive conversion does not provide complete results. The attributive system uses unambiguous tags for verbs which prevent the homonymous understanding – the verb form *pomohly* (they helped) with the tag `k5eAaPmAgFnP`

clearly refers to a plural feminine subject (the attribute `gF`). However, the positional system allows using of ambiguous tags – the verb form *pomohly* has the tag `VpTP---XR-AA---` where the third position T applies to the feminine as well as the masculine inanimate gender. On that account, the `--posttags` switch gives the tag `k5mAnPgIgFnPp.mReA` to the output which the SET is not able to process (double attribute `gIgF`) and during the analysis, SET works only with the first attribute `gI`. At the end, the SET announces a false report.

If SET could work with all possible interpretations gained from a tag, we suppose whole analysis will get better results. With this supposition, we manually fixed the way how the `--posttags` switch evaluates tag matching. After that, SET takes into account more possible interpretations if they are recorded in a tag. Subsequent testing reduced the number of false reports (FP) from 48 to 12 which rapidly increased the accuracy (See the table 5). No other results were affected by this adjustment.

3.4 Colloquial expressions in written texts

The last part of testing was focused on stylistic. Details about this module were presented in [5] and [12]. With regard to the type of the rules, there were not as many problems as in other modules because these rules are not so dependent on morphological analysis.

The biggest issue was colloquial expressions in written text - e.g. *hezkej nábytek* (a nice furniture), where MorphoDiTa had no match. It is caused by the `--posttags` switch: in majka atributive system there is a part of the tag containing `wH`, which means conversational/colloquial [6]. However, the `--posttags` switch does not convert this part of the tag to MorphoDiTa's fifteenth position, which holds stylistic variant [2].

Other rules have more or less same results as presented in [5] and [12], so we do not consider them important to mention.

4 Conclusion

Testing did not prove that the MorphoDiTa system would arrange a big difference in results. MorphoDiTa mildly wins in automatic detection of zeugma and detects errors in subject-predicate agreement with better accuracy. However, the majka analyzer with the DESAMB tagger provides better precision and recall in general evaluation of the automatic insertion of missing commas. The detection of multiple sentence members and the detection of errors in subject-predicate agreement indicate that MorphoDiTa deals with case homonymy better than DESAMB tagger.

The using of the MorphoDiTa system could be advantageous since the system works with the maintained dictionary and is updated on a regular basis. Therefore, it would be practical to tune up the `--posttags` switch which will be able to convert ambiguous positional tags. Nevertheless, we also see the room for improvement of tools that we have used up to know and which would

lead to satisfying outcome: 1. to implement linguistic rules that would improve disambiguation of the DESAMB tagger or to develop/find better tagger; 2. to update the dictionary which is used by the majka analyzer.

In conclusion, we would like to mention a paradox that partly affects our work: An excellent automatic grammar checker needs the best possible output of the morphological analysis. But in case an analyzer should provide the best analysis, it requires a text with a minimum of mistakes.

Acknowledgements This work was supported by the project of specific research *Čeština v jednotě synchronie a diachronie* (Czech language in unity of synchrony and diachrony; project no. MUNI/A/1061/2018) and by the Technology Agency of the Czech Republic under the project TL02000146.

References

1. Behún, D.: Lingea Grammaticon – přísný strážce jazyka českého, https://www.interval.cz/clanky/lingea-grammaticon-prisny-strazce-jazyka-ceskeho/
2. Cvrček, V., Richterová, O.: seznamy:tagy – příručka ČNK (2017), http://wiki.korpus.cz/doku.php?id=seznamy:tagy&rev=1497540816
3. Geržová, H.: Automatická detekce negramatických větných konstrukcí pro češtinu. Master's thesis, Masaryk University, Faculty of Arts, Brno (2019 [cit 2019-10-30]), Available from: https://is.muni.cz/th/fuz2y/
4. Hajič, J., Hlaváčová, J.: MorfFlex CZ (2013), http://hdl.handle.net/11858/00-097C-0000-0015-A780-9, LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University
5. Hlaváčková, D., Hrabalová, B., Machura, J., Masopustová, M., Mrkývka, V., Valíčková, M., Žižková, H.: New online proofreader for czech. In: Slavonic Natural Language Processing in the 21st Century. p. 56–69. Tribun, Brno (in printing)
6. Jakubíček, M., Kovář, V., Šmerk, P.: Czech morphological tagset revisited. Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2011 pp. 29–43 (2011)
7. Karlík, P.: Zeugma. In: CzechEncy - Nový encyklopedický slovník češtiny (2017), https://www.czechency.org/slovnik/ZEUGMA
8. Kovář, V., Horák, A., Jakubíček, M.: Syntactic analysis using finite patterns: A new parsing system for czech. In: Language and Technology Conference. pp. 161–171. Springer (2011)
9. Kovář, V., Machura, J., Zemková, K., Rott, M.: Evaluation and improvements in punctuation detection for czech. In: International Conference on Text, Speech, and Dialogue. pp. 287–294. Springer (2016)
10. Kovář, V.: Partial Grammar Checking for Czech Using the SET Parser, pp. 308–314. Springer (2014). https://doi.org/10.1007/978-3-10816-2_38
11. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text tokenisation using unitok. In: RASLAN. pp. 71–75 (2014)
12. Novotná, M., Masopustová, M.: Using syntax analyser set as a grammar checker for czech. In: Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2018. pp. 9–14 (2018)

13. Pala, K., Rychlý, P., Smrž, P.: Desam—annotated corpus for czech. In: International Conference on Current Trends in Theory and Practice of Computer Science. pp. 523–530. Springer (1997)
14. Petkevič, V.: Kontrola české gramatiky (český grammar checker). Studie z aplikované lingvistiky-Studies in Applied Linguistics **5**(2), 48–66 (2014)
15. Pořízka, P.: Tvorba korpusů a vytěžování jazykových dat: metody, modely, nástroje. Vydavatelství Filozofické fakulty Univerzity Palackého v Olomouci (2014)
16. Šmerk, P.: Towards morphological disambiguation of Czech. Ph.D. thesis, Masaryk University, Faculty of Informatics, Brno (2007)
17. Šmerk, P.: Fast morphological analysis of czech. In: Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2007. pp. 13–16 (2009)
18. 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 (2014)
19. Suchomel, V.: cstenten17, a recent czech web corpus. In: Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2018. pp. 111–123 (2018)

Implementing an Old Czech Word Forms Generator

Ondřej Svoboda

Czech Language Institute of the Czech Academy of Sciences
Valentinská 91/1, 116 46 Praha 1, Czech Republic
svoboda@ujc.cas.cz

Abstract. This paper presents a word forms generator for Old Czech, originally covering common nouns but extended to produce forms of other POS, with their specifics (adverbs, prepositions, verbs). After describing the background, it gives an account of the development process and dives into the steps the generator performs. Two fundamental components of the generator are then shown in detail. Apart from the current status, future steps are also suggested.

Keywords: Old Czech, word forms generator, lemmatization, implementation

1 Preface: formal description of declension of Old Czech common nouns

In 2017, the first part of formal description of Old Czech[1] inflectional morphology was conceived (including the lexicon): Pavlína Synková concluded her dissertation [4] on declension of Old Czech (OC) common nouns, created, from the very start, with algorithmic implementation in mind.

The intended use of the implementation was (and still is) automatic lemmatization of "Old Czech text bank" (from now on, called "the corpus"), an ever-growing collection of manually transcribed and edited Old Czech manuscripts and prints (the "reference material")[2] hosted at *Vokabulář webový* ("VW")[3] since 2006.

1.1 Origins of the formal description

Building on detailed description of declension patterns of common nouns given by modern grammars, Synková first used a corpus-based tool to validate, extend and formalize the description with regards to the reference material[4] and

[1] Old Czech period spans from around 1150 to 1500.

[2] In contrast to transliteration, the text is rendered in modern orthography and error-corrected while preserving features of OC, such as the phoneme *ě* (yat) in *cěsta* ("path, road, journey"; Modern Czech: *cesta*).

[3] https://vokabular.ujc.cas.cz

[4] She mostly worked with a non-public, bigger version of the corpus (containing "alpha quality" transcriptions) but consulted the manuscripts many times (in doubt or when handling rare declension patterns), improving the corpus' quality in the process.

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 15–21, 2019.

second, she carefully assigned the formalized declension patterns (from now on, "paradigms") to almost 30,000 common noun headwords extracted from modern dictionaries also hosted at VW.

To handle variation in *inflectional bases* she introduced a few kinds of "alternation", assigned to individual headwords. E.g., in an E—∅ alternation, when comparing *o**kn**-o* ("window") and *o**ken*** (in genitive plural), "kn" (symbolically, `KK`) alternates with "ken" (`KeK`; `K` stands for a consonant).

She covered language change by developing a set of targeted search & replace rules ("sound changes") to transform early OC surface forms into late OC forms. To post-process raw productions into (orthographically) correct forms, she designed further rules ("formal changes"), allowing the paradigms and headwords to contain only the linguistically relevant, compact information.

E.g., *ušiech* (a local plural form of *ucho*, "ear") is produced from **uch-iech*, composed of an inflectional base and a regular ending, by turning "chie" into "šie". Afterwards, the form develops into *uších* ("ie" > "í"). Using the two rules avoids, respectively, the need for, possibly, a special case in alternation (dual forms use an *uš-* base; including also certain plural forms would be complicated), and also the need for a more modern ending (*-ích*).

2 Continuous extension & implementation

Since later in 2017, work has been underway on both sides: a word forms generator was written in .NET/C# to build a morphological database, eventually used to annotate the corpus with a home-grown tagger (in Python), handling specifics of the corpus[5].

To drive initial development and contribute to quality and stability, a list of more than 100 headwords (at least one for each paradigm) was compiled to check for correct as well as defective productions, helping to spot regressions.

A simple web interface[6] was created to provide interactive access to the generator, inspect word forms (arranged in tables) and their properties, and discover errors not yet covered by the regression test.

After "proving the concept" with a working implementation covering common nouns, further headwords and paradigms have been added, prompting changes at the generator's side. Additions so far include all immutable POS (prepositions, conjunctions, particles, and interjections), adverbs (both gradable and ungradable), and a few large conjugation classes of verbs.

2.1 Features of headwords

With nouns, alternations of the inflectional base were introduced, as well as a simple paradigm constraint (singular- or plural-only). Adverbs can declare

[5] Such as using occasional manual lemmatization by editors of the source manuscripts to partially disambiguate otherwise highly ambiguous automatic annotation. (No OC training corpus is currently available to me although one existed in the past [1].)

[6] `https://ridics.ujc.cas.cz/nlp/word-forms/`

(often suppletive) gradated forms in place of a regular paradigm. Similarly, prepositions list the governed cases and vocalized ones specify vowels to attach (*k*, *ke*, *ku*).

Verbs provide their aspect and will require multiple types of alternation (e.g. vowel quantity and modification of the final consonant cluster) to apply simultaneously.

3 From a headword to word forms

This section describes the process of generating word forms from a headword, given its POS or paradigm ID, and other properties, mentioned above, as input.

3.1 Paradigm setup

After the paradigm is either looked up by its ID, constructed at runtime (for prepositions and irregular adverbs), or an invariant one is used, it is accomodated to the headword and its constraints. For example, *húsle* ("fiddle") selects only plural endings of an otherwise complete paradigm *kost*.

Terminology: Since endings of some paradigms also come with suffixes, a more general concept will be used in the following text. "Terminations" *proper* refer to endings with suffixes. Generalized "terminations" can, in addition, carry further morphs such as the superlative prefix (*naj-*), various forms of the negative prefix, and even suppletive inflectional bases (in verbs like *býti*, "to be").

Sound changes are applied to the paradigm, creating (proper) terminations more recent than the initial, early 14th century ones. The changes will affect the additional morphs specified above, in the near future.

3.2 "Stemming"

Before an inflectional base can be retrieved, special, "stemming"-only terminations are also created when performing sound changes, to account for a pre-1300 *lě* > *le* sound change, and related ones. For *húsle*, the closest suitable termination in `PL.NOM` is *-ě*, which cannot be removed directly. For this purpose, an *-e* termination is developed, requiring "l" before it. The "sound change" used here is written with regex notation: `(?<=l)ě > e`.

After the (longest) stemming termination is removed, the resulting raw base undergoes a round of supplementary formal changes to suit the paradigm. These affect headwords like *mládě* ("a young [being]"). After losing the *-ě* the base-final consonant is softened (> `mláď`) in order to produce e.g. a plural form *mláďata*.

3.3 Alternations

At this stage, alternations of the inflectional base are handled, distributing terminations of the headword's paradigm between resulting multiple variant bases. For *okno*, the ending *-o* is first removed from the initial base "okn".

The headword's E—∅ alternation is specified as `E0- zero_endings-KeK / nonzero_endings-KK`. First, the *-o* ending is matched against the `nonzero_endings` selector, with the respective `KK` pattern. Second, the two placeholders resolve to "k" and "n", and the remainder of the base ("o", an immutable "prefix") is also stored.

By applying the obtained consonants to the `KK` and `KeK` templates and appending the results to the stored "prefix", two variant bases (*okn-* and *oken-*) form. Finally, all non-zero endings are associated to the *okn-* base, while the *oken-* base is joined by a *-*∅ `PL.GEN` ending.

3.4 Further base changes

Sound changes are applied to the inflectional bases next. In addition, a *ne-* prefix is added for "negatable POS" (currently, verbs). For verbs like *obalovati*, this results in a chain of *obal-* > *vobal-* > *nevobal-* bases.

3.5 Emission

For each base, early OC word forms are created first by joining prefixes with the base and suffixes (all of various origin). Afterwards, sound changes apply to the boundary between the base and suffixes (both proper ones like *mořě* ("sea") > *moře*, and auxiliary ones like **húslě* > *húsle*), and subsequently to the whole word forms: *púšče* ("deserts") > *púště*.

Throughout the whole process, extra information is attached to terminations, inflectional bases, and the resulting word forms, such as sound changes applied to the respective units. Each word form is thus aware of its "origin", allowing to produce a tag made up of common grammatical categories. In the future, an additional tag distinguishing the word form from others with an identical grammatical tag (following ideas of [3]) should be possible.

4 Components of the generator

Adjusting to the need to generate word forms of various POS, the initial noun-specific implementation has grown a few interesting features and components.

4.1 Paradigms module

The generator is capable of modeling paradigms of any Old Czech POS in a single framework, while catering to their needs. Starting with a fixed two-level structure sufficient to model paradigms of nouns (three numbers with

seven cases each), the framework was generalized to house case-governing prepositions, gradable adverbs (with a capability to define regular creation of superlatives from comparatives), and highly inflected verbs.

An example below shows parts of a paradigm of verbs conjugating like *pračĕvati*. Ancestors of `<termination/>` nodes describe the structure (of uneven depth) common to all Old Czech verbs, defined in a separate meta file.

```
<present type="PRES/I">
  <singular number="SG/S">
    <firstPerson person="1/1">
    ...
<supine type="SUP/U" matchAspect="IPFV"/>
<participle type="PART">
  <nt subtype="NT" likeParadigm="declension" type="/S"
      substrate="verb.6.kupovati"/>
  ...
  <l subtype="L" type="/A">
    <singular number="SG/S">
      <termination gender="M">
        <stemSuffix>ěva</stemSuffix>
        <participleSuffix>l</participleSuffix>
        <ending>0</ending>
      </termination>
```

Some of the structure nodes' attributes carry two values. In `@type`, `@subtype`, `@number`, and `@person`, the first part is a "glossing abbreviation"[7] instrumental in referring to a paradigm node by a path. For instance, `PRES.SG.1` is used in verbs such as *prositi* to change the root-final consonant from "s" to "š", producing *prošu*, and `PL` is used to select a part of a paradigm in plural-only nouns. Following the slash, the second part is a value of the Czech attributive tagset [2], giving (partial) tags `nSp1mI`, `mU`, and `gMnSmA` for the above example.

The `@substrate` attributes allow (nominal and verbal) paradigms to inherit terminations from (parts of) other paradigms. The `@matchAspect` attribute guards against generating supines for perfective verbs.

The `<termination/>` nodes are found in leaf nodes of the structure and contain a complete set of morphs to be attached to (or even replace) an inflectional base, already introduced in 3.1. This allows to generate *nenie*, *nejnie*, and *nénie* (`PRES.SG.1.NEG`) for *býti* or use full, supplied forms for complex cases like *týden* ("week"): *téhodne*.

4.2 Sound changes engine

This component has been shown to be essential to several stages of word forms generation. So far, sound changes have targeted terminations ("proper" only,

[7] `https://en.wikipedia.org/wiki/List_of_glossing_abbreviations`

not "generalized"), inflectional bases (both "raw" and post-alternation ones), the boundary between the base and suffixes, and whole word forms.

Up to now, about 50 out of 100 changes defined in the dissertation have been enabled in the generator.

As an example, the `u-i-change`, representing a pair of sound changes written as *'u* > *'i* and *'ú* > *'í*, is accompanied by two kinds of information, relevant to the generator, or only to users. It lists its targets (inflectional bases, base—suffix boundaries, terminations), a contemporary/related change (`o-yat-change`), the approximate time span it was in effect (2nd to 3rd fourth of the 14th century), and finally its specification (here in a redacted form), using a regex look-ahead to block it from applying on the *uo* digraph:

`Ku(?!o) > Ki` and `Kú > Kí` where

`K = ž,š,č,ř,j,ď,ť,ň,ḃ,ḟ,ľ,ṁ,ṗ,ṡ,v́,ź,c`

It could also define a short value to use in a "variant/mutation" [3] tag to distinguish the input form (*zeḿu*) from an (intermediate) output (*zeḿi*).

The lower bound of the time information could be used in two ways: to attach an estimate datation ("not before") to the targets, and enforce a chronological order when applying sound changes.

5 Conclusion

This article presented a generator capable of producing Old Czech word forms on the grounds of a list of headwords (with POS-specific properties), a linguistically adequate repertoire of inflectional paradigms (as compact as possible), and a set of sound changes targeted at concrete morphemes.

During the process, the generator keeps track of the origins of each word form, enabling it to attach useful, rich information to its output.

Acknowledgements This work was fully supported by RIDICS ("Research Infrastructure for Diachronic Czech Studies"), a project LM2015081 funded by the Czech Ministry of Education, Youth and Sports of the Czech Republic.

References

1. Hana, J., Lehečka, B., Feldman, A., Černá, A., Oliva, K.: Building a corpus of old czech. In: Proceedings of the 8th International Conference on Language Resources and Evaluation, LREC. p. 1–7. European Language Resources Association, Istanbul (2012), `https://msuweb.montclair.edu/~feldmana/publications/2012-morph-lrec.pdf`
2. Jakubíček, M., Kovář, V., Šmerk, P.: Czech morphological tagset revisited. In: Proceedings of Recent Advances in Slavonic Natural Language Processing. p. 29–42. Tribun EU, Brno (2011), `https://nlp.fi.muni.cz/raslan/raslan11.pdf#page=37`
3. Osolsobě, K., Hlaváčová, J., Petkevič, V., Šimandl, J., Svášek, M.: Nová automatická morfologická analýza češtiny [the new automatic morphological analysis of czech]. Naše řeč, AV ČR, Ústav pro jazyk český **100**(2), 225–234 (2017)

4. Synková, P.: Popis staročeské apelativní deklinace (se zřetelem k automatické morfologické analýze textů Staročeské textové banky) [Description of Old Czech Common Nouns Declension (with regard to Automatic Morphological Analysis of Texts in Old Czech Text Bank)]. Ph.d. thesis, Charles University, Faculty of Arts, Institute of Czech Language and Theory of Communication, Praha (2017), http://hdl.handle.net/20.500.11956/86563

Neural Tagger for Czech Language: Capturing Linguistic Phenomena in Web Corpora

Zuzana Nevěřilová and Marie Stará

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

Abstract. We propose a new tagger for the Czech language and particularly for the tagset used for annotation of corpora of the TenTen family. The tagger is based on neural networks with pretrained word embeddings. We selected the newest Czech Web corpus of the TenTen family as training data, but we removed sentences with phenomena that were often annotated incorrectly. We let the tagger to learn the annotation of these phenomena on its own. We also experimented with the recognition of multi-word expressions since this information can support the correct tagging.
We evaluated the tagger on 6,950 sentences (84,023 tokens) from the `cstenten17` corpus and achieved 75.25% accuracy when compared by tags. When compared by attributes, we achieved 91.62% accuracy; the accuracy of POS tag prediction is 96.5%.

Keywords: Czech Tagger, Multi-word Expressions, Pretrained Word Embeddings

1 Introduction

Currently, the processing pipeline for Czech Web corpora of the TenTen family [2] uses the tagger `desamb` [7], based on the morphological analyser `majka` [11], the guesser, and inductive logical programming. This approach has been sufficient for many texts that follow the grammatical, syntactic, and orthographic rules of the language. With the rise of Web corpora, a more flexible solution is desired. The basic requirements comprise:

1. adaptability to typos/incorrect orthography
2. adaptability to neologisms
3. ability to distinguish foreign language injections
4. ability to annotate foreign proper nouns
5. adaptability to newly appearing syntactic patterns
6. ability to recognize multi-word expressions (MWEs)

While the requirements 1, 2, and 4 are to a large extent fulfilled by the current pipeline, requirement 3 has been solved in several works, and requirements 5 and 6 are currently not solved for the Czech language. We propose a new tagger

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 23–32, 2019.

based on neural networks that is able to learn from the current Web corpus of Czech and thus is able to reflect the change in the language use.

This work aims to be one of the steps towards an adaptable MWE-aware tagger for Czech.

1.1 Paper Outline

Section 2 describes current taggers for Czech language, Section 3 describes the training examples selection process. In Section 4, we provide detailed information on the neural network architecture and parameters. Section 5 provides evaluation results and error analysis. In Section 6, we plan further work in the area.

2 Related Work

Historically, for Czech, two different tagsets are used, unfortunately not easily convertible. The tagset used for annotation of corpora of the TenTen family is an attributional system used in the tagger `desamb` [7] and the morphological analyser `majka` [11]. A detailed description of the tagset is in [3].

For the positional tagset, the most widely used tagger MorphoDiTa was developed [8]. The authors report 95.75% accuracy in POS tagging. MorphoDiTa replaces its predecessor Morče based on the same algorithm.

The focus of another tagger, MUMULS [10], is in verbal MWE identification. The tagger annotates POS and MWEs. The authors report that token F1 is 73% on Czech language, which is only one among 10 languages the tagger is able to process.

3 Training Data

As training data for our tagger, we selected first million sentences from the Web corpus `cstenten17` [9]. In the version `mj2` we used, the corpus was tagged using `majka+desamb` pipeline `v2`[1].

From previous versions of the `cstenten` (formerly `cztenten`) corpus, some problematic phenomena were known. In our previous work [5], we identified the incorrect annotation of inter-lingual homographs (words that exist in different languages but have a different function). For example, the word *top* is an English noun or adjective, while in Czech, it is an imperative form of *to drawn* or *to stoke up*.

The second group of problems is caused by the fact that the guesser is intended to be used for Czech out-of-vocabulary words (OOVs). Its performance is much lower in the case of foreign names (person names, brand names, place names, etc.). A nice example of such annotation is the word *Wikipedia* often annotated as plural genitive (probably because of its Latin-like ending).

[1] Further information in `https://www.sketchengine.eu/cstenten-czech-corpus/`

The third group of problematic annotation results in guessed lemmata for some groups of Czech OOVs that never appear as words in the corpus. There can be a case that a word is never used in its base form, but it is very likely that these lemmata were simply guessed incorrectly. Examples of such annotations are words such as *šedat* or *mývalit*, incorrectly recognized as verbs instead of adjectives *šedý*, *mývalí*.

Because of the systematic nature of the incorrect annotations of the phenomena mentioned above, we filtered out sentences containing these phenomena. We created a blacklist composed of OOV proper names, interlingual homographs, and guessed lemmata that never appear in the corpus as words. We kept 606,351 sentences from which we selected a random sample.

4 Training the Model

Tagging is a sequence processing task, therefore a usual neural tagger is composed of the input layer, embedding layer, one or several hidden recurrent layers, and the output layer. The usual scheme is depicted in Figure 1. The input sequence I represents the maximum number of tokens s, embedding layers encodes the tokens into vectors, the output layer O represents tags for tokens, its length c is the number of possible tags.

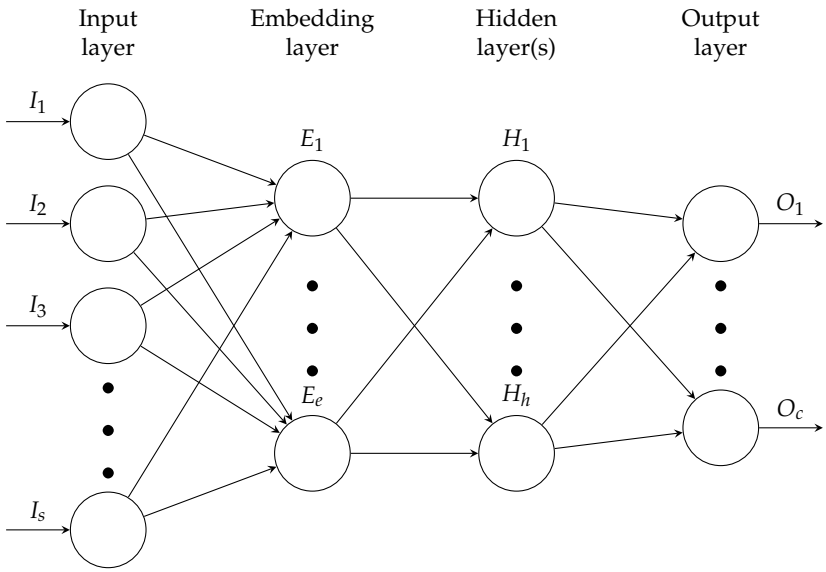


Fig. 1: Usual architecture for a tagger. The input sequence has length s (the number of tokens), the output layer length c is the number of possible tags.

4.1 Tagset Encoding

Since Czech taggers have to annotate many grammatical categories (that have many possible values), the number of possible tags is too large. In the Universal dependencies, the `ajka` tagset contains 2,176 tags[2]. For comparison, the Penn Tree Bank tagset in Universal Dependencies has 48 different tags[3]. Our cleaned sample of 606k sentences uses 1,537 different tags. For experimental settings, we removed the following attributes:

- stylistic subclassification (attribute `w`)
- subclassification of pronouns (attribute `x`)
- subclassification of adverbs (attribute `y`)
- stylistic subclassification (attribute `w`)
- punctuation subclassification (attribute `x`)
- verb aspect (the attribute `a`)

After this tagset reduction, we split every tag into attributes and reformulate the problem. In typical cases, tagging is a classification problem with one-hot output vector. In our setting, each token belongs to n classes where $n = 0, ..., m$ and m is the maximum number of attributes in one tag. For example, a noun in nominative singular masculine inanimate (tag `k1gInSc1`) has four classes (noun, nominative, singular, masculine inanimate). Using this technique, we reduce the number of tags to 44. On the other hand, the implementation is slightly more complicated, since the output is not a sequence of one-hot vectors but a sequence of multi-hot vectors.

4.2 Pretrained Embeddings

Pretrained embeddings are very popular since they can be easily reused to capture the semantic relations between words. Using pretrained embeddings usually improves the tagger results significantly. The usual procedure is to set pretrained embeddings as input weights of the embedding layer E (see Figure 1) and set the layer non-trainable. The advantage is ease of implementation and the final model size, the disadvantage is the OOV problem, since the neural network contains only embeddings of the training examples.

Embedding models can be calculated in several ways, and pretrained models are available for many languages, mostly trained on Wikipedia data. We use `fasttext` [1] since it contains subword information which supports language modeling tasks significantly.

Inspired by [6], we incorporate the `fasttext` model directly into the input layer. We did not use the `Embedding` layer provided by TensorFlow Keras[4] because we implemented the embedding layer on our own. The consequence is that every input sequence of tokens has to be converted into a sequence of

[2] `https://universaldependencies.org/tagset-conversion/cs-ajka-uposf.html`
[3] `https://universaldependencies.org/tagset-conversion/en-penn-uposf.html`
[4] `https://www.tensorflow.org/guide/keras/overview`

embeddings using the same `fasttext` model. On the other hand, the advantage of this approach is a massive improvement of classification since we completely avoid the OOV problem.

4.3 Neural Network Architecture and Parameters

We set up a neural network with two bidirectional long short term memory network (Bi-LSTM) layers, both with spatial dropout. We limited the maximum sentence length to 20 tokens. For longer sentences, the prediction has to use a sliding window. The size of the two Bi-LSTM layers are 512 and 128, respectively, the size of the output layer is 44 (the number of possible attributes in tags). A scheme of the architecture is in Figure 2.

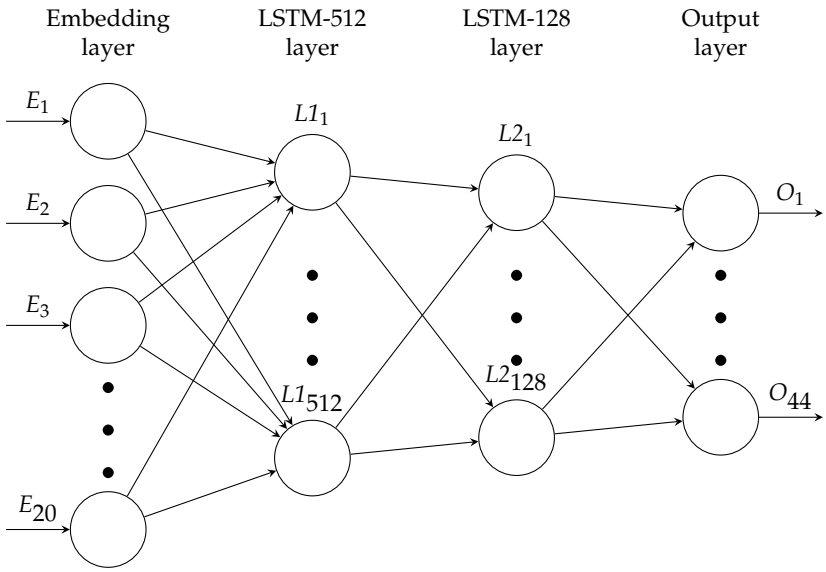


Fig. 2: Architecture of the tagger: the input layer already contains word embeddings, two Bi-LSTM layers follow with the dropout layers (not in the schema).

We used sigmoid activations for all layers and Adam optimizer together with binary cross-entropy loss. The input sequences were padded by zero vectors. We experimented with inserting tag distribution as input weights of the last layer, but those did not improve the network results.

The training took 9 epochs on 180,000 samples split into 171,000/19,000 train/test data. The measured accuracy ended up at 99.38%, and the measured validation loss went down to 1.70%. The results are so optimistic mainly because

of the zero padding since it is easy for the network to learn the tags for zero vectors.

The output layer simply performs rounding, so all values equal to or greater than 0.5 are converted to tag attributes. Furthermore, post-processing selects only one attribute in case several values of the same attribute have values over 0.5. For example, if the tagger predicts 0.56 for attribute `c1` and 0.65 for attribute `c4`, the post-processing selects `c4`. We do not force the tagger to predict all possible attribute values of the tag, so it can happen that the tagger predicts no attribute.

The model size is 18MB, but for prediction, the `fasttext` model (7GB) is needed in addition. According to the authors of `fasttext`, the model size can be shrunk using quantization. We did not solve the issue yet.

5 Evaluation

We performed a detailed quantitative evaluation of the tagger on 6,950 sentences not present in the training set but present in the cleaned data. The tag accuracy was 75.25% if measured by an exact match. Since the tagger is not forced to predict all possible attributes, we also measured the submatch accuracy, i.e., the case where the predicted tag is contained in the true tag. We achieved 87.62% submatch accuracy. Finally, we also measured the match between the attributes of the predicted and true tags. Here, we achieved 91.62% accuracy. Accuracy on the POS tag (the `k` tag) was 96,5%, the tagger did not predict any tag in 1,2% cases.

5.1 Quantitative Error Analysis

Not all tagging errors are equally serious. In the cases we mentioned in Section 3, the error severity is caused mainly by incorrect part-of-speech (POS) tagging since the POS tag is used in further statistical computations. Another type of errors, known in the corpus annotation, is incorrect gender and incorrect case (mainly distinction between nominative and accusative). These types of errors can also affect further processing, e.g., syntactic analysis.

In Figures 3 and 4, we provide confusion matrices calculated separately for attributes POS (`k`), gender (`g`), number (`n`), and case (`c`). The numbers represent hundreds of examples from the test data.

From confusion matrices, we can observe that misclassification of the POS tag does not occur very often. The most confused classes are nouns and interjections, nouns and adjectives, and adjectives and verbs. The noun-adjective uncertainty can be caused by nominalization (substantivization of adjectives) as well as by English borrowings with noun modifiers. The adjective-verb uncertainty originates in the similarity between adjectives and N-type past participles.

The second important observation concerns grammatical cases, where the nominative (case 1) and accusative (case 4) are often interchanged. This type of error is known already from the current annotation pipeline. Its solution is, therefore, very challenging since we know the training data is not clean enough.

From Figure 4, it can be seen number and especially gender are the most difficult attributes to annotate. Surprisingly, the dominant gender is feminine, and also the highest number of confusion is between feminine and the other genders. For verb attributes (grammatical mood and person) as well as for adjective and adverb attribute degree, the confusion matrices do not show much interesting information, so we do not provide them.

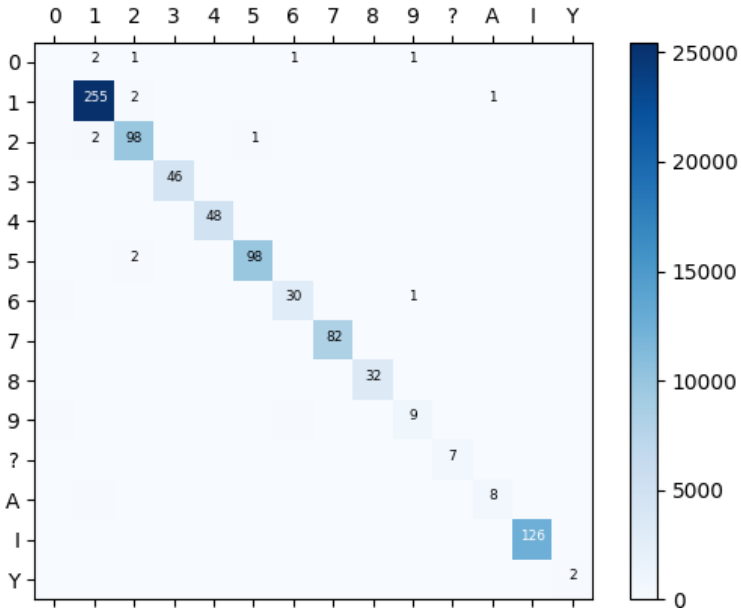


Fig. 3: Confusion matrix for the POS (attribute `k`).

5.2 Multi-word Expression Discovery

Using `fasttext`, we experimented with MWE discovery. We observed that in case of words a and b that are part of an MWE, their vectors are more similar to the vector of concatenated word $a + b$ than random words. We, therefore, calculate similarity matrices for each pair of neighboring words. If the sum of all elements in the similarity matrix is above a threshold t, we consider the pair to be an MWE. We only excluded punctuation from this calculation; however, we want to experiment with the grammatical information obtained by the tagger. Currently, we use the thresholds between $t = 7.1$ and $t = 7.3$. In the examples below, the identified MWEs are *Sulfid bismutitý* (meaning *bismuth(III) sulfide*), with threshold $t = 7.3$. With $t = 7.1$, the identified MWEs are also *jsou zveřejněny* (meaning *are published*), *and Cons*, *Hitch Hiking*, *The Wall*.

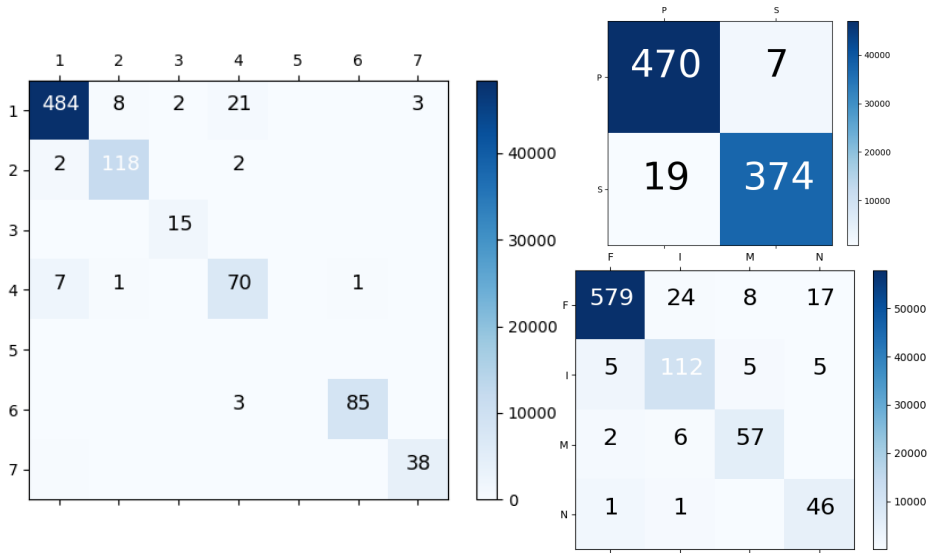


Fig. 4: Confusion matrices for the case, number, and gender (attributes `c`, `n`, and `g` respectively. N.B. that nouns, adjectives, pronouns, and numerals distinguish cases, number is an attribute also for verbs.

5.3 Manual Evaluation on Small Sample

The aim of the work is to outperform the current annotation pipeline, especially in the case of foreign words and inter-lingual homographs. To test this, we manually annotated a few random sentences from the corpus `bulky` [4]. The corpus is a subcorpus of the corpus `cztenten12` containing problematic phenomena such as foreign names and inter-lingual homographs. Figure 5 presents the outputs on five sentences. It can be seen that a serious error – incorrect POS tag – occurs 4× in case `desamb`, twice in case of our solution. Most other errors occur in the gender and case attributes. Some differences in the tagging are subject to discussion, for example, how to annotate foreign words or MWEs that play the role of a noun (e.g., the word *and* inside a multi-word named entity).

6 Conclusion and Future Work

We present a new tagger for Czech trained on the Web corpus `cztenten17` annotated using the `majka` tagset. We show that the neural approach is a promising direction, especially when combined with pretrained word embeddings containing subword information. The current POS tag accuracy is slightly above the results published for the tagger MorphoDiTa; however, the comparison was not made on the same data set.

	Do	vyhledávacího	pole	zadejte	Texas	HoldEm	Poker	.[5]
desamb	k7c2	k2gNnSc2d1	k1gNnSc2	k5mRp2nP	**kA**	k1gInS**c7**	k1gInSc1	kIx.
our	k7c2	k2gNnSc2*	k1gNnSc2	k5*p2*	k1gInSc1	k1gInSc1	k1gInSc1	kIx.

	Výsledky	jsou	zveřejněny	v	časopise	BMC	Biology	.[6]
desamb	k1gInPc1	k5mIp3nP	k5mNgInP	k7c6	k1gInSc6	kA	k1**gMnPc4**	kIx.
our	k1gInPc1	k5mIp3nP	k5mNgInP	k7c6	k1gInSc6	kA	k1nS	kIx.

	Lady	Murasaki	ztělesnila	čínská	herečka	Gong	Li[7]
desamb	k1gFnPc4	k1gFnPc4	k5mAgFnS	k2gFnSc1d1	k1gFnSc1	k1**gI**nS**c4**	**k8xS**
our	k1gFnP**c1**	k1gFnP**c1**	k5mAgFnS	k2gFnSc1d1	k1gFnSc1	k1gFnS	k1gFnS

	Album	The	Pros	and	Cons	of	Hitch	Hiking	…
desamb	k1gNnSc4	k1gFnS**c2**	k1gFnP**c2**	k?	k1gInS**c1**	k?	k1gInS**c1**	k1gInSc4	
our	k1gNnS**c1**	k1nS**c1**	k1nS**c1**	k1nS	k1nS**c1**	kA	k1gInS**c1**	k1gInS**c1**	
…	psal	Waters	v	roce	1978	současně	s	The	Wall[8]
desamb	k5mA**gI**nS	k1**gI**nSc1	k7c6	k1gInSc6	k4	k6d1	k7c7	k1gFnSc7	k1gFnP**c2**
our	k5mAgMnS	k1**gI**nSc1	k7c6	k1gInSc6	k4	k6d1	k7c7	k1nS	k1nS**c1**

	Sulfid	bismutitý	Bi2S3	je	tmavěhnědá	(	pokud	se …
desamb	k1gInSc1	k2g**Mn**Sc1d1	k1g**M**nSc1	k5mIp3nS	k2**gNnPc4**d1	kIxX	k8	k3c4
our	k1gInSc1	**k1gI**nSc1	k1nSc1	k5mIp3nS	k2*nSc1d1	kIxX	k8	k3c4

	připravuje	srážením	se	sirovodíkem	)	nebo	šedá	látka[9]
desamb	k5mIp3nS	k1gNnSc7	**k3c4**	k1gInSc7	kIxX	k8	**k5mIp3nS**	k1gFnSc1
our	k5mIp3nS	k1*nSc7	**k3c4**	k1gInSc7	kIxX	k8	k2gFnSc1d1	k1gFnSc1

Fig. 5: Example comparison between `desamb` and our tagger. Bold attributes are for sure incorrect, other differences can be discussed. The stars mean that our tagger did not provide any attribute and there should be some.

In the near future, we aim to retrain the model with the full tagset. We reduced the tagset mainly because we needed the model to fit into memory. We plan to implement generators to reduce the data needed to be loaded in memory at once.

For future work, we plan to add lemmatizer to the neural network. With the lemmatizer, we want to focus on borrowings and neologisms, since these words are often processed incorrectly by the guesser.

The cleanness of the training data is among known issues, so we plan to solve the problem with incorrect case annotation (mainly nominative and accusative). We will possibly incorporate `majka` outputs and semantic constraints to distinguish these two cases.

Another planned direction is the ability of the tagger to identify foreign injections and annotate MWEs. These two phenomena are related, and we hope

[5] Enter Texas HoldEm Poker into the search field.

[6] The results are published in the journal BMC Biology.

[7] Lady Murasaki was played by Chinese actor Gong Li.

[8] Waters wrote the album The Pros and Cons of Hitch Hiking in 1978 together with The Wall.

[9] Bismuth III sulfide Bi2S3 is dark brown (if prepared by precipitation with hydrogen sulfide) or gray substance.

to solve them using pretrained embeddings, grammatical information, and possibly corpus statistics.

Last but not least, we plan to provide the tagger as a service. We also plan to publish a model based on Universal Dependencies since this widely used tagset will allow a fair evaluation.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin infrastructure LM2015071 and OP VVV project CZ.02.1.01/0.0/0.0/16_013/0001781.

References

1. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)
2. Jakubíček, M., Kilgarriff, 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/
3. 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)
4. Pelikánová, Z., Nevěřilová, Z.: czTenTen12 v9 subcorpus of problematic phenomena (2018), http://hdl.handle.net/11234/1-2822, LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University
5. Pelikánová, Z., Nevěřilová, Z.: Corpus annotation pipeline for non-standard texts. In: Sojka, P., Horák, A., Kopeček, I., Pala, K. (eds.) Text, Speech, and Dialogue. pp. 295–303. Springer International Publishing, Cham (2018)
6. Schumacher, M.: Using FastText models (not vectors) for robust embeddings (2 2018), https://www.kaggle.com/mschumacher/using-fasttext-models-for-robust-embeddings
7. Šmerk, P.: Unsupervised learning of rules for morphological disambiguation. In: Sojka, P., Kopeček, I., Pala, K. (eds.) Text, Speech and Dialogue. pp. 211–216. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
8. 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. Association for Computational Linguistics, Baltimore, Maryland (June 2014), http://www.aclweb.org/anthology/P/P14/P14-5003.pdf
9. Suchomel, V.: csTenTen17, a recent czech web corpus. Twelveth Workshop on Recent Advances in Slavonic Natural Language Processing pp. 111–123 (2018)
10. Variš, D., Klyueva, N.: Improving a neural-based tagger for multiword expressions identification. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki, Japan (May 2018), https://www.aclweb.org/anthology/L18-1401
11. Šmerk, P.: Fast morphological analysis of Czech. In: Proceedings of the Raslan Workshop 2009. Masarykova univerzita, Brno (2009)

Evaluation and Error Analysis of Rule-based Paraphrase Generation for Czech

Veronika Burgerová and Aleš Horák

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00, Brno, Czech republic
{xburger, hales}@fi.muni.cz

Abstract. In this paper, we present the experiments and evaluation of previously developed rule-based paraphrasing system for the Czech language. The system offers several interconnected modules that allow to generate paraphrases of an input sentence based on various criteria such as the Czech WordNet hierarchy, word-ordering rules or anaphora resolution. We have evaluated each module's accuracy and we offer a detailed analysis of the results as well as concrete proposals for improvements.

Keywords: paraphrasing, rule-based, game with a purpose, Czech

1 Introduction

The possibility to programmatically identify or generate paraphrases of an input text allows a plethora of practical natural language processing applications such as machine translation [3], text summarization [15], or semantic interpretation of phrases [6,2]. The techniques for paraphrase generation range from explainable rule-based or thesaurus-based methods [14] to unsupervised approaches usually inspired by machine translation solutions as a monolingual translation [10,9].

In this paper, we evaluate the current results of a previously published rule-base paraphrasing system for Czech [7], which was explicated in a game-with-a-purpose application named Watsonson. We offer a detailed analysis of the results of each of Watsonson's modules and also propose their further development.

2 The Watsonson Project

In general, the task of automatic quality evaluation of a generated sentence is quite difficult. If reference results prepared by human annotators are available, the evaluation can proceed by comparative measures. Without such gold standard datasets, the evaluation mostly relies on human judgement. However, the manual annotation and evaluation of a large set of sentences can be expensive. In the Watsonson project, the paraphrasing results are evaluated in a crowdsourcing approach in the form of a *game with a purpose* (GWAP [1]).

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 33–39, 2019.

Table 1: Example paraphrases generated by individual Watsonson modules.

module	sentence
Input	Pejsci si chtěli hrát s **dětmi**, ale žádné děti venku nenašli. (Dogs wanted to play with **children** but they found no children outside.)
wordhyp	Pejsci si chtěli hrát s **lidmi**. (Dogs wanted to play with **people**.)
wordnet	Pejsci si chtěli hrát s **děcky**. (Dogs wanted to play with **kids**.)
wordorder	Nenašli **oni** žádné děti venku. (**They** did not find any children outside.)
aara	**Pejsci** nenašli žádné děti venku. (**Dogs** found no children outside.)
verbinfer	**Neobjevili** žádné děti venku. (They **discovered** no children outside.)

The project uses existing tools such as a morphological analyzer or a syntactic parser and comes up with rule-based procedures to generate paraphrases. The independence of these modules allows us to use and evaluate each module separately.

A detailed description of the Watsonson project in available in [8]. We briefly introduce its five modules we have experimentally evaluated. Example paraphrases generated by the particular modules are presented in Table 1.

2.1 Wordnet and Wordhyp

The *wordnet* module uses data from the Czech WordNet [11] for synonym replacement. Recursively, the words can be replaced by their hypernyms, which is the task of the related the *wordhyp* module. The modules currently do not employ any word sense identification technique to distinguish word senses, which is why wrong paraphrases can be generated.

2.2 Word order

Considering the flexibility of Czech word order, sentence constituents can be reordered in many combinations which still form a correct Czech sentence. This *word order* generates phrases with all possible orders of the sentence constituents.

2.3 Aara

The *Aara* module implements a partial anaphora resolution system. This system resolves zero subjects and replaces pronominal objects or subjects by their co-referent antecedents. In the module, such phrases are generated and offered for annotation.

2.4 Verbinfer

The Czech verb valency lexicon VerbaLex [4] is used in this module, which uses the verb frame inference of three types: equality, effect and precondition to transform the phrases. For instance, *be sad* might be an effect of *get lost*. Besides generating paraphrases, this module can also result in new facts.

Table 2: Statistics of the input testing dataset.

# of sentences	40
# of clauses	97
minimum sentence length	2
maximum sentence length	40
# of words	530
# of pronouns	63
# of named entities	10

3 Experiments and Evaluation

Although the Watsonson users evaluate the sentences given by the paraphrase generator, we can not always distinguish which parts of the system lead to the good evaluations and which ones cause errors. Also, the evaluation of a sentence may be subjective and the decision whether the sentence is correct or not might be ambiguous in some cases.

That is why, the presented evaluation experiment worked with individual modules only and the input paraphrases were processed and by each module separately and tested for correctness.

3.1 Preprocessing

The first phase of the experiment consisted in taking 40 Czech sentences of various complexity. The sentences were either simple made up phrases or they were extracted from Czech children tales. Detailed statistics of this dataset are displayed in Table 2.

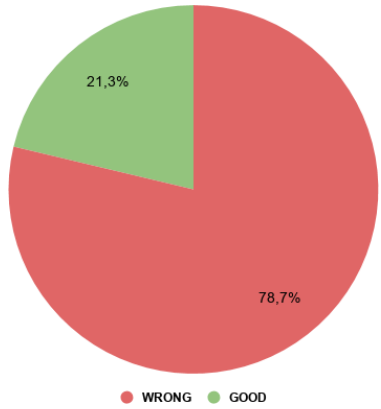


Fig. 1: The total score of the five modules.

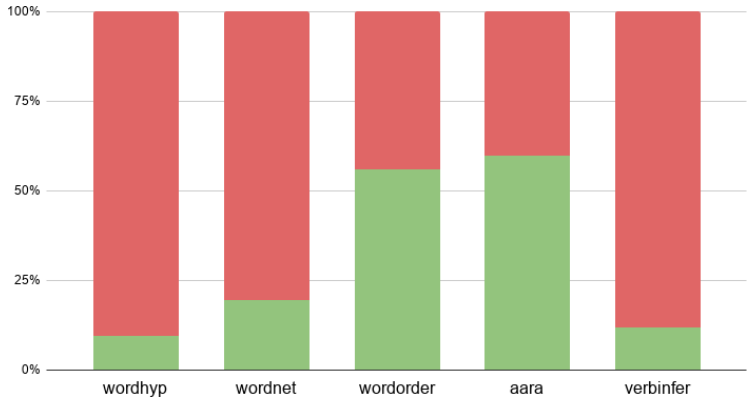


Fig. 2: The individual scores of each module.

As the individual modules rely on morphological and syntactic annotations, the sentences we first processed by the annotating tools: morphological analyzer/generator *majka* [12], morphological tagger *desamb* [13] and the syntactic parser *SET* [5].

The preprocessing pipeline results with tagged and parsed sentences were stored in the JSON format and server as an input to each of the evaluated modules. In total, the paraphrasing modules generated 1,514 new sentences based on the 40 original statements.

3.2 Evaluation

Within the evaluation, all the generated paraphrases were manually annotated. At first, each sentence was marked whether it is or is not a good paraphrase of its input.

As can be seen in Figure 1, only 21.3 % of the 1514 paraphrases were evaluated as good paraphrases. Comparing this score with the score of the paraphrases

Table 3: The statistics of the paraphrasing results per module.

module	# of good paraphrases		# of bad paraphrases		total # of paraphrases	
wordhyp	52	← 9%	505	←91%	557	↓37%
wordnet	95	←19%	393	←81%	488	↓32%
wordorder	93	←56%	73	←44%	166	↓11%
aara	58	←60%	39	←40%	97	↓ 6%
verbinfer	24	←12%	182	←88%	206	↓14%
total	322	21%	1,192	79%	1,514	100%

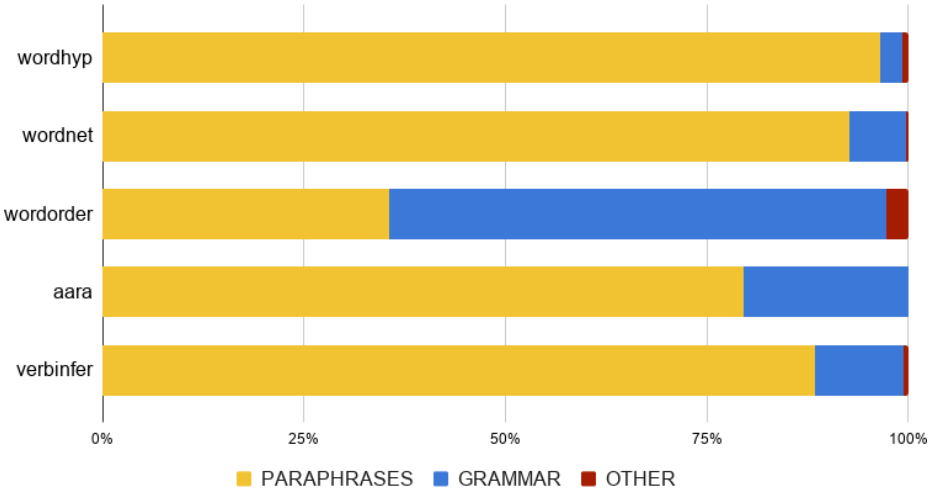


Fig. 3: The results of the error analysis applied on each of the modules.

evaluated by Watsonson users, which is more than 55 %, the total score is lower than expected.

Focusing on the modules' results separately in Figure 2, we are able to see which modules are beneficial to the paraphrase generation process and which ones take the total score down. The score ratios and numbers of sentences generated by the particular modules are presented in Table 3.

3.3 Error analysis

After the basic evaluation, the next step focussed on detailed analysis of the errors in the generation process. The errors were classified in the following three categories:

- **Incorrect paraphrase**. This means that the generated sentence does not make sense at all or does not follow the meaning of the original sentence.
- **Incorrect grammar**. The sentence meaning is synonymous to the meaning of the original sentence and it would make a good paraphrase, but it is not grammatically correct.
- **Other**. Unspecified type of an error not fitting any of the types mentioned above.

The results of this detailed error analysis for each module are presented in Figure 3. Most of the errors are caused by incorrect paraphrasing, however, we can observe that a significant percentage of errors is grammatical, especially in the *wordorder* module.

In the last part of the analysis, we focused on the most common errors that occurred in the results and tried to find out the source of such errors.

As we have shown, most of the incorrect paraphrases were generated by the *wordhyp*, *wordnet* and *verbinfer* modules. All of these modules generate paraphrases on the basis of replacing words in a sentence by words with similar meaning. Nevertheless, these meanings often do not fit in the given context.

On the contrary, most of the errors made by the *wordorder* module, were caused by forming an ungrammatical sentence. Nevertheless, the overall score of this module is comparatively high due to the fact that the Czech word order is very flexible, but it is not completely free.

After the individual evaluation of each module separately, we have analysed the errors that occurred repeatedly across all the modules and identified the main reasons of them:

1. **Prepositions**. We have noticed several paraphrases that were either missing a preposition where it was needed or occurred with incorrect or redundant preposition.
2. **Dependencies**. In a lot of paraphrases, we observed incorrect dependencies among the parts of a sentence, for instance, object generated as subject etc.

Further analysis of the primary source if these errors revealed that a significant part of these errors was being caused by errors in the syntactic parsing phase.

4 Conclusions and Future Work

We have presented a detailed evaluation and error analysis of five paraphrase generation modules of the Watsonson project. The analysis showed the most problematic sources of errors in the generation process and helped to pave road for further improvements of the system.

New modules are planned to provide new types of paraphrasing methods such as replacing other sentence constituents than nouns and verbs or transforming the active voice to passive and vice versa.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin infrastructure LM2015071.

References

1. von Ahn, L.: Games with a purpose. Computer **39**(6), 92–94 (2006)
2. Bollegala, D., Shutova, E.: Metaphor interpretation using paraphrases extracted from the web. PloS one **8**(9), e74304 (2013)
3. Callison-Burch, C., Koehn, P., Osborne, M.: Improved statistical machine translation using paraphrases. In: Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics. pp. 17–24. Association for Computational Linguistics (2006)

4. Hlaváčková, D., Horák, A., Kadlec, V.: Exploitation of the VerbaLex verb valency lexicon in the syntactic analysis of Czech. In: International Conference on Text, Speech and Dialogue, TSD 2006. pp. 79–85. Springer (2006)
5. Jakubíček, M., Horák, A., Kovář, V.: Mining phrases from syntactic analysis. In: International Conference on Text, Speech and Dialogue, TSD 2009. pp. 124–130. Springer (2009)
6. Nakov, P.I., Hearst, M.A.: Semantic interpretation of noun compounds using verbal and other paraphrases. ACM Transactions on Speech and Language Processing (TSLP) **10**(3), 13 (2013)
7. Nevěřilová, Z.: Paraphrase and textual entailment generation. In: International Conference on Text, Speech, and Dialogue, TSD 2014. pp. 293–300. Springer (2014)
8. Nevěřilová, Z.: Annotation game for textual entailment evaluation. In: Gelbukh, A.F. (ed.) Proceedings of CICLing. LNCS, vol. 8403, pp. 340–350. Springer, Heidelberg (2014)
9. Prakash, A., Hasan, S.A., Lee, K., Datla, V., Qadir, A., Liu, J., Farri, O.: Neural paraphrase generation with stacked residual lstm networks. arXiv preprint arXiv:1610.03098 (2016)
10. Quirk, C., Brockett, C., Dolan, W.: Monolingual machine translation for paraphrase generation. In: Proceedings of the 2004 conference on empirical methods in natural language processing. pp. 142–149 (2004)
11. Rambousek, A., Pala, K., Tukačová, S.: Overview and Future of Czech Wordnet. In: McCrae, J.P., Bond, F., Buitelaar, P., Cimiano, P., 4, T.D., Gracia, J., Kernerman, I., Ponsoda, E.M., Ordan, N., Piasecki, M. (eds.) LDK Workshops: OntoLex, TIAD and Challenges for Wordnets. pp. 146–151. CEUR-WS.org, Galway, Ireland (2017), `http://ceur-ws.org/Vol-1899/`
12. Šmerk, P.: Fast Morphological Analysis of Czech. In: Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2009. pp. 13–16 (2009)
13. Šmerk, P.: K počítačové morfologické analýze češtiny (in Czech, Towards Computational Morphological Analysis of Czech). Ph.D. thesis, Faculty of Informatics, Masaryk University (2010)
14. Zhao, S., Lan, X., Liu, T., Li, S.: Application-driven statistical paraphrase generation. 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: Volume 2-Volume 2. pp. 834–842. Association for Computational Linguistics (2009)
15. Zhou, L., Lin, C.Y., Munteanu, D.S., Hovy, E.: Paraeval: Using paraphrases to evaluate summaries automatically. In: Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics. pp. 447–454. Association for Computational Linguistics (2006)

Part II

NLP Applications

Recent Advancements of the New Online Proofreader of Czech

Vojtěch Mrkývka

Faculty of Arts, Masaryk University
Arne Nováka 1, 602 00 Brno, Czech Republic
mrkyvka@phil.muni.cz

Abstract. On the previous RASLAN workshop, the basis of the new online proofreader for the Czech language was presented. This paper describes the current status quo of this tool as well as describes changes necessary due to alterations of the assignment.

Keywords: grammar checker, text analysis, proofreading, Czech

1 Introduction

The new online proofreader is being developed at the Masaryk University (with the help from external experts) since 2018. The motivation to do so is the parallel development of separate rulesets for finding different types of mistakes in the Czech language[3] (for example spelling, commas or agreement) using SET analyser[1] as well as lack of real proofreading tool for the online environment. The previous paper, *Towards the New Czech Grammar-checker*[1][2], compared the new proofreader with the proofreading capabilities of Microsoft Office Word as described in *Kontrola české gramatiky (český grammar checker)* by Vladimír Petkevič[4], seeing differences in approach to mistakes and their highlighting as well as to the licencing of the final product.

2 The original approach

The first version of the proofreader was implemented as part of the TinyMCE online text processor.[2] It consisted of separate modules with limited mutual dependency (see fig. 1), but with strong connections to the editor itself.[3] The major part was written in JavaScript with connection to Python backend when necessary.

[1] There was a dispute whether to use expression grammar-checker, Grammar Checker, proofreader or something else as authors of different components used this inconsistently. In May of 2019, the consensus was reached and the Online Proofreader is used since. Sorry for the inconvenience.

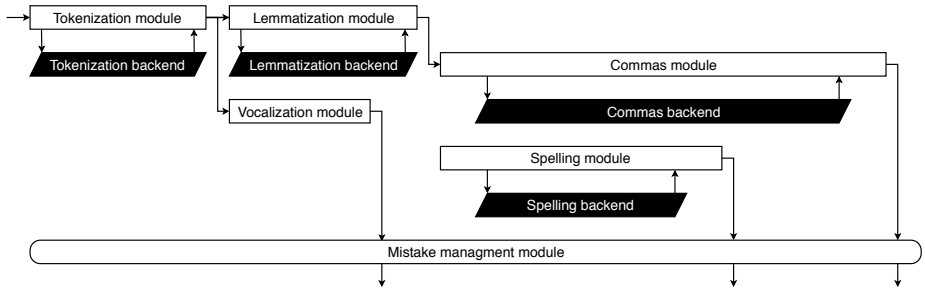
[2] `https://www.tiny.cloud/`

[3] For a more detailed description read already mentioned *Towards the New Czech Grammar-checker*.

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 43–47, 2019.

The main advantage of this approach was that the different components (existing and new) could have been independently developed and also the new versions of these components could be independently released. This, however, led to problems when the need of improvement touched some of the core functionalities (such as the mistake manager). Additionally, this approach was unpleasant from the end-user point of view as the installation and maintaining process was not very intuitive.

Fig. 1: The workflow diagram of the first version of the proofreader. Components depicted in white are separate TinyMCE modules. The horizontal axis suggests time.



3 The JavaScript-first approach

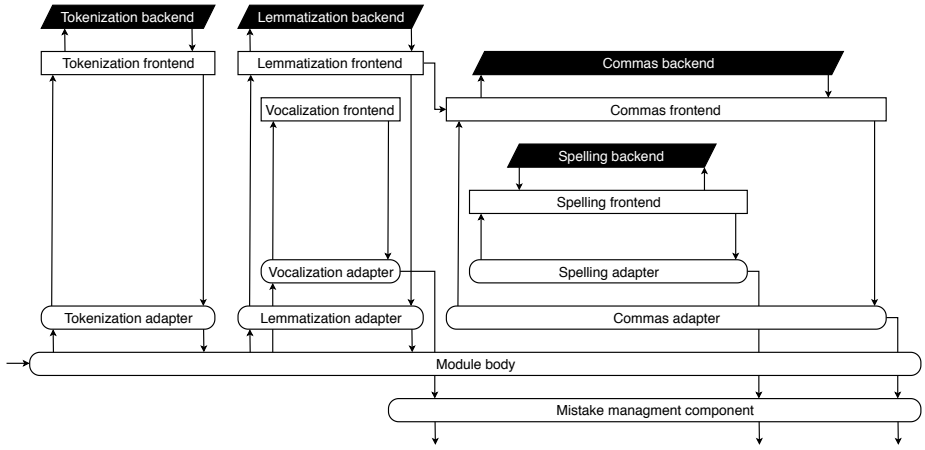
Learning from the previous mistakes, the second version, although created as separate JavaScript files, worked independently on the TinyMCE. This approach allowed to adapt the proofreader for other environments. Additionally, when used in one, the compatible versions of components could have been used and distributed as a single module (see fig. 2).

Because of the (at the time unknown) demand to use the proofreader in other environments than the browser applications, the development of the second version had to be cancelled as it would be unsustainable with these conditions.

4 The Python API approach

Currently in development is the third version of the proofreader created as an application programming interface written in Python. This provides a unified access point to be used in browser-based and non-browser based applications alike. The API-based approach also creates natural way to test the proofreader with large portion of texts as it does provide single machine-readable output. Along with the new approach, new features are presented as well as solutions to the existing problems.

Fig. 2: The workflow diagram of the second version of the proofreader. Elements depicted in white are components of the single module (where *frontend* means universal interface and *adapter* adaptation of the universal data to the TinyMCE specific environment). The horizontal axis suggests time.



4.1 Alternative tagger

The new version of the proofreader includes the alternative option to lemmatisation and tagging – *MorphoDiTa*[6] – and currently is being tested as an alternative to the combination of *majka* (lemmatiser and tagger)[8] and *DESAMB*[7] (disambiguator) which was used in previous versions.

The main problem with implementation is that the tagsets used by majka and MorphoDiTa differ in formal properties (attributive vs positional) as well as in encoded information. It follows that these two tagsets are not mutually convertible without losing some of the information provided. Conversion, however, has to be done since rulesets created for majority of proofreading components are dependent on majka's attributive tags. Similarly, some of the components need to have the source text separated into sentences. This information is not provided by MorphoDiTa. For the testing purposes, I used majka to help with sentence limits knowing it will need to be altered in the future to achieve better performance. A secondary problem is the attachment of additional information to the lemma such as lemma category (link to the PDT – Prague dependency treebank); thus the final output is for example `jet-1_^(pohybovat_se,_ne_však_chůzí)` instead of simple `jet`. This was resolved with a simple regular expression.

4.2 Asynchronicity and performance

Due to the single-query nature of the API, it is not possible to have mistakes displayed gradually depending on the time different components finish their

processing, but there has to be single output. However, an advantage of this approach is that there is higher pressure to make individual proofreading components faster. Besides, although the output can be only singular, the idea of asynchronicity and parallel processing of the output was preserved using the *asyncio* module, which was included into Python standard library in version 3.4.

Performance-wise there is a higher focus on the inner workings of Python, to achieve better speed. For example information about tagged words are stored as *named tuple* (from the *collections* module of the Python standard library) instead of widely-used *dictionary* keeping the data smaller memory-wise, but preserving the same readability.[5] See the following example for the sentence *Jak se máš?*:

```
[TaggedWord(word='Jak', lemma='jak', tag='k6eAd1', type='WORD'),
 TaggedWord(word=' ', lemma='', tag='', type='WHITESPACE'),
 TaggedWord(word='se', lemma='se', tag='k3xPyFc4', type='WORD'),
 TaggedWord(word=' ', lemma='', tag='', type='WHITESPACE'),
 TaggedWord(word='máš', lemma='mít',
            tag='k5eAaImIp2nS', type='WORD'),
 TaggedWord(word='?', lemma='?',
            tag='kIx.', type='MULTICHAR_PUNCTUATION')]
```

To keep the way to further optimisation open, the crucial components, such as the mistake manager, are wrapped inside the class, providing consistent output despite alterations of their inner workings.

4.3 Updates on proofreading components

As the structure of the proofreader changed heavily from the first version, the proofreading components had to be reworked as well. As the majority of these components used SET analyser as part of their doing, the adaptation was a relatively simple alteration of the backend part of the original component. Some of the components were updated with improved rulesets to provide better results. However, as adapting these components is currently ongoing or recently finished, there are no complex testing results to be published yet. There are also new components; for example, the component focused on detection of non-grammatical constructions such as attraction or blending errors.

5 Conclusion

This paper follows the current status of the proofreading tool developed at Masaryk University as well as the overall progress made during the recent year following the previous description of the project in *Towards the New Czech Grammar-checker*. Multiple issues, such as independence on the specific text processor, were resolved already with other planned to be resolved in the proximate future. Although the development brings many obstacles, overcoming them is necessary to bring the project to the successful end.

Acknowledgements This work was supported by the project of specific research *Čeština v jednotě synchronie a diachronie* (Czech language in unity of synchrony and diachrony; project no. MUNI/A/1061/2018) and by the Technology Agency of the Czech Republic under the project TL02000146.

References

1. 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)
2. Mrkývka, V.: Towards the New Czech Grammar-checker. In: Horák, A., Rychlý, P., Rambousek, A. (eds.) Proceedings of the Twelfth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2018. pp. 3–8. Masaryk University, Brno (2009)
3. Novotná, M., Masopustová, M.: Using Syntax Analyser SET as a Grammar Checker for Czech. In: Horák, A., Rychlý, P., Rambousek, A. (eds.) Proceedings of the Twelfth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2018. pp. 9–14. Masaryk University, Brno (2009)
4. Petkevič, V.: Kontrola české gramatiky (český grammar checker). Studie z aplikované lingvistiky-Studies in Applied Linguistics **5**(2), 48–66 (2014)
5. Ramalho, L.: Fluent Python, p. 96. O'Reilly, Sebastopol (2015)
6. 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. Association for Computational Linguistics, Baltimore, Maryland (June 2014), `http://www.aclweb.org/anthology/P/P14/P14-5003.pdf`
7. Šmerk, P.: Towards morphological disambiguation of Czech. Ph.D. thesis, Masaryk University, Brno (2007)
8. Šmerk, P.: Fast Morphological Analysis of Czech. In: Sojka, P., Horák, A. (eds.) Proceedings of Third Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2009. pp. 6–9. Masaryk University, Brno (2009)

Approximate String Matching for Detecting Keywords in Scanned Business Documents

Thi Hien Ha

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a 602 00 Brno, Czech Republic
462259@mail.muni.cz

Abstract. Optical Character Recognition (OCR) is achieving higher accuracy. However, to decrease error rate down to zero is still a human desire. This paper presents an approximate string matching method using weighted edit distance for searching keywords in OCR-ed business documents. The evaluation on a Czech invoice dataset shows that the method can detect a significant part of erroneous keywords.

Keywords: approximate string matching, Levenshtein distance, weighted edit distance, OCR, invoice

1 Introduction

Business documents, different from other types of documents, are obligation to have a predefined set of data which is usually specified by keywords. Therefore, localization of keywords in the document plays an important role in document processing. However, scanned business documents potentially involve OCR errors which exact match cannot solve.

A deep statistical analysis of OCR errors covering five aspects, based on four different English document collections was described in [7]. They find out that among three most common edit operation types in OCR errors (insertion, deletion, and substitution), substitution is much more frequent than the others with average of 51.6%, and most of OCR errors can be corrected just by single operation types (total percentage of three single operations is 77.02%). The analysis also results in detailed statistics of standard mapping and non standard mapping, which is valuable to create character confusion matrix, one of the most important sources for generating and ranking candidates in error correction.

OCR post-processing aiming at fixing the residual errors in OCR-ed text. The approaches for this problem can be categorized into dictionary-based and context-based types. Dictionary and character n-gram are often used in the former to detect and correct isolated word errors whereas the latter consider grammatical and semantics context of the errors. This usually relies on word n-grams and language modeling [3,2].

However, business documents such as invoices has different characteristics in comparison with data using in those methods. Firstly, invoices are written

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 49–54, 2019.

in short chunk instead of fully grammatical text. Secondly, non-words are frequent in invoices, involving entities names, and almost all of data fields such as invoice/order number, item codes, account number. Approximate string matching is string matching of a pattern in a text that allows errors in one or both of them [6]. It very soon became a basic tool for correcting misspelling words in written text, and then, in text retrieval since exact string matching was not enough due to large text collection, more heterogeneous, and more error-prone.

In this paper, we use approximate string matching based on a weighted edit distance to search for keywords in scanned business documents. The method is evaluated on a Czech invoice dataset.

2 Method

2.1 Problem definition

The problem of approximate string matching is defined as follow:
Let Σ be a finite alphabet; $|\Sigma| = \sigma$; Σ^* is the set of all strings over Σ
Let $T \in \Sigma^*$ be a text of length n; $|T| = n$
Let $P \in \Sigma^*$ be a pattern of length m; $|P| = m$
Let d: $\Sigma^* \times \Sigma^* \rightarrow \Re$ be a distance function.
Let $k \in \Re$ be the maximum number of error allowed.
The problem is given T, P, k and d(.), return the set of all the substrings of T: $T_{i..j}$ such that $d(T_{i..j}, P) \leq$ k.

The distance d(x, y) between two strings x and y is the minimal cost of a sequence of operations that transform x into y. The operations are a finite set of rules ($\delta(z, w)$). In most of applications, the set of operations is limited to:

- *Insertion*: $\delta(\varepsilon, a)$: inserting the letter a
- *Deletion*: $\delta(a, \varepsilon)$: deleting the letter a
- *Substitution*: $\delta(a, b); a \neq b$: substituting a by b
- *Transposition*: $\delta(ab, ba); a \neq b$: swap the adjacent letters a and b.

One of the most commonly used distance function is Levenshtein [1], also called edit distance. Edit distance $d(x, y)$ between two strings x, y is the minimal number of insertions, deletions, and substitutions to transform x into y. The distance is symmetric, i.e. $d(x, y) = d(y, x)$. In the simplified definition, all the operations cost 1. Therefore, $0 \leq d(x, y) \leq max(|x|, |y|)$.

2.2 Weighted edit distance

Detecting keywords from OCRed documents faces at least two problems. The first problem to take into account is OCR errors. There are both standard mapping 1:1 (one character is mis-recognized into another character, e.g "email" and "emall") and non-standard mapping 1:n or n: 1 (e.g "rn" and "m"). However, the dominant portion of OCR errors is standard mapping. More example of common errors can be seen in Table 1. Another type of OCR errors is incorrect

word boundary, including incorrect split (i.e wrongly splitting one word into two or more strings) and run-on error (i.e inaccurately putting two or more words together).

Table 1: Common pairs of OCR errors

Character 1	Character 2	Character 1	Character 2
b	h	n	r
c	o	O	0
c	e	r	i
C	(	r	t
f	t	s	5
f	l	v	y
i	l	z	2
l	|	z	s
l	1	y	g
l	ť	m	n

Beside OCR errors, the other problem comes from the language characteristics. Modern Czech orthographic system is diacritical. The acute accent and *háček* are added to Latin letters, such as "á", "í", "ě", "č". Moreover, grammatically, Czech is inflectional, like other Slavic languages. The missing or excessive diacritics, using different endings, either by typing errors or OCR errors make the problem worse.

Being aware of those problems, we set different costs for operations. For those substitutions of common OCR errors, or substitutions between pairs of short and long vowels, with or without háček (from now on, they are mentioned as common OCR errors), we set a much cheaper cost (e.g 0.1) than the normal one (i.e 1). The lower cost is also set for inserting or deleting of spaces and punctuation. We call this as a weighted edit distance function.

$$\delta(a,b) = \begin{cases} 0 & \text{if } a = b \\ 0.1 & \text{if } \begin{bmatrix} (a = \varepsilon \text{ and b is punctuation}) \\ \text{or } (b = \varepsilon \text{ and a is punctuation}) \\ \text{or } (a,b) \text{ is a common pair of OCR errors} \end{bmatrix} \\ 1 & \text{otherwise} \end{cases}$$

Because insertion and deletion have the same weight, the distance function is still symmetric.

2.3 Algorithms

We call $\alpha = k/m$ the error ratio. Since we can make the pattern match at any position in the text by performing m substitutions, $0 \leq k \leq m$ (reminding $m = |P|$). Therefore, $0 \leq \alpha \leq 1$.

In the problem of searching for keywords in the text allowing an error ratio α, P is the keyword. We propose a filtering algorithm involving following steps:

- Get the longest consecutive common substring of the keyword and the text. If the ratio between length of the substring and length of the keyword is less than a third, then return no approximate match found.
- Extend the substring in the text to get the longest substring with smallest edit distance.
- If the ratio between weighted edit distance of the keyword and new substring and length of the keyword does not exceed α, then return the substring. Otherwise, return no approximate match found.

Weighted edit distance function using dynamic programming:

```
function Weighted_distance(s1[0..m−1], s2[0..n−1]):
  float prev_row[0..n]
  float  cur_row[0..n]
  prev_row[0] = 0
  for i from 1 to n:
      prev_row[i] = prev_row[0] + δ(ε,s2[i−1])
  for i from 1 to m:
      cur_row[0] = prev_row[0] + δ(s1[i−1],ε)
      for j from 1 to n:
          insertions = prev_row[j] + δ(ε, s2[j−1])
          deletions  = cur_row[j−1] + δ(s1[i−1], ε)
          substitutions = prev_row[j−1] +
                          δ(s1[i−1], s2[j−1])
          cur_row[j] = min(insertions, deletions,
                           substitutions)
      prev_row = cur_row
  return prev_row[−1]
```

For Levenshtein or edit distance function, we just need to replace δ function by 1. Because weighted edit distance function is slower than normal edit distance, in the the second step, we use original edit distance and only calculate weighted edit distance at the final step.

There has been a significant number of research in implementation of approximate string matching to reduce time and space complexities. Methods based on finite state automation promise to be much faster than dynamic programming and can be computed in linear time [8,4,5].

3 Experiments

The dataset contains 50 Czech scanned invoices. After using an open source OCR to get the text, we run two different modules to detect a set of given keywords. The former is exact matching using regular expression. The latter uses proposed

Table 2: Analysis of kewords detected by approximate string matching but missed by exact matching

Types of error	Number of keywords	in %
OCR errors	52	30.4%
Diacritics	32	18.7%
Inflection	44	25.7%
False possitive	43	25.1%
Total	171	100%

approximate string matching. Then, we compare to see how many keywords the approximate string matching detected that exact matching did not.

Keywords for 36 fields are detected, including invoice number, invoice date, order number, order date, due date, payment date, and so on. The length of keywords varies from 1 (e.g invoice number: "č") to 43 (e.g payment date: "datum uskutečnění zdanitelného plnění").

The threshold for the error rate is set to 0.15 in the experiment. This means, for example with a keyword of length 6, the distance allowed is $6 \times 0.15 = 0.9$, i.e it allows only common OCR errors.

The result is summarized in Table 2. In total 171 keywords detected by approximate string matching but missing by exact match, 30.4% are because of OCR errors in one characters (e.g "C'Slo" instead of "Cislo", "Odběrate!" instead of "Odběratel", "e-mall" instead of "e-mail"), 18.7% caused by missing or excess of diacritics (e.g in "č", "ě", "í"). The different endings by inflection, for instance "objednávka" and "objednávky", are the reason for 25.7%. The last 25.1% are caused by close keywords. Let take the title "invoice" and the field "invoice number" as an example. One of keywords for this field is "Daňový doklad č" which has only a space and one character ("č", abbreviation of "číslo", means "number") differ from the title "Daňový doklad". Therefore, the distance ($d = 1.1/15 \approx 0.07$) is less than the threshold (0.15), resulting title is marked as keyword. This error can be filtered out by checking if the annotated substring is also marked as title. In fact, in many invoices, there is an invoice number on the same line of title without the word "number" accompanied. Therefore, title in these cases is the signpost to extract the invoice number, the same role as a keyword.

Besides, we notice that there are 20 keywords containing OCR errors are still missing by approximate string matching. These errors are not in the given OCR common errors. Almost half of them are standard mapping, e.g "o" becomes "g", "I" becomes "'", or "ft" becomes "||". The other half are caused by non standard mapping, such as "čt" becomes "d", or "j" becomes "f'".

4 Conclusion

In this paper, we have described approximate string matching approach based on a weighted edit distance for detecting keywords in scanned business documents.

The experiment shows that this method adapts pretty well to erroneous text and inflectional languages. The future work will focus on a complete list of common errors and implementation using finite state automation.

Acknowledgements This work has been partly supported by Konica Minolta Business Solution Czech within the OCR Miner project. This publication was written with the support of the Specific University Research provided by the Ministry of Education, Youth and Sports of the Czech Republic.

References

1. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady. vol. 10, pp. 707–710 (1966)
2. Llobet, R., Cerdan-Navarro, J., Perez-Cortes, J., Arlandis, J.: Ocr post-processing using weighted finite-state transducers. In: 2010 20th International Conference on Pattern Recognition. pp. 2021–2024 (Aug 2010). https://doi.org/10.1109/ICPR.2010.498
3. Mei, J., Islam, A., Moh'd, A., Wu, Y., Milios, E.: Statistical learning for ocr error correction. Information Processing & Management **54**(6), 874–887 (2018)
4. Mihov, S., Schulz, K.U.: Fast approximate search in large dictionaries. Computational Linguistics **30**(4), 451–477 (2004)
5. Mitankin, P.: Universal levenshtein automata. building and properties. Sofia University St. Kliment Ohridski (2005)
6. Navarro, G.: A guided tour to approximate string matching. ACM computing surveys (CSUR) **33**(1), 31–88 (2001)
7. Nguyen, T., Jatowt, A., Coustaty, M., Nguyen, N., Doucet, A.: Deep statistical analysis of ocr errors for effective post-ocr processing. In: 2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL). pp. 29–38 (June 2019). https://doi.org/10.1109/JCDL.2019.00015
8. Schulz, K.U., Mihov, S.: Fast string correction with levenshtein automata. International Journal on Document Analysis and Recognition **5**(1), 67–85 (Nov 2002). https://doi.org/10.1007/s10032-002-0082-8, `https://doi.org/10.1007/s10032-002-0082-8`

Structured Information Extraction from Pharmaceutical Records

Michaela Bamburová and Zuzana Nevěřilová

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

Abstract. The paper presents an iterative approach to understanding semi-structured or unstructured tabular data with pharmaceutical records. The task is to split records with entities such as drug name, dosage strength, dosage form, and package size into the appropriate columns. The data is provided by many suppliers, and so it is very diverse in terms of structure. Some of the records are easy to parse using regular expressions; others are difficult and need advanced methods. We used regular expressions for the easy-to-parse data and conditional random fields for the more complex records. We iteratively extend the training data set using the above methods together with manual corrections. Currently, the F1 score for correct classification into 5 classes is 95%.

Keywords: structured information extraction, table understanding, entity recognition

1 Introduction

National authorities for drug control and administration publish the drug data including prices in order to provide pharmaceutical companies, patient organizations, and other interested parties the most recent data. Price Monitor, a product of COGVIO[1] company, is a data processing and analysis tool for planning, payer negotiations, and price management of drugs with always up-to-date information. It integrates more than 100 global public data sources of drug-related data and offers various insights and analysis for market access and pricing teams. For comparing product prices, it is necessary not only to know what products are similar by dosage strength but also by dosage form, package size, distributor, or country. This information is present in the provided data; however, not always in an easy-to-parse form.

Figure 1 illustrates a typical case. One drug is provided in several dosage strengths, and the comparison of prices has to take these aspects into account.

Some suppliers provide already structured data. This data is easy to parse into the appropriate columns such as *name*, *dosage strength*, *dosage form*, *package size*, *company*, *price*, *ATC*[2] and many others. On the other hand, many suppliers

[1] `https://www.cogvio.com/`

[2] Anatomical Therapeutic Chemical is a classification system for active substances.

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 55–62, 2019.

Country	Brand name	Company	ATC	Active Substance	Dosage Form	Dosage strength	Package	
CZ SCAU	ADEMPAS	BAYER AG, LEVERKUSEN	C02KX05	RIOCIGUAT	TBL FLM	1,5MG	42	EUR 1,210.07 Ex-F
CZ SCAU	ADEMPAS	BAYER AG, LEVERKUSEN	C02KX05	RIOCIGUAT	TBL FLM	2,5MG	84	EUR 2,609.42 Ex-F

Fig. 1: Two examples of the same drug but in different dosage strengths and package sizes.

provide all the data in one column, and it is often difficult to identify and parse the data precisely, which is crucial for further analysis and reports.

Figure 2 illustrates one already structured drug data and one with part of the data in the same column.

Country	Brand name	Company	ATC	Active Substance	Dosage Form	Dosage strength	Package
LU Legilux	ABILIFY CPR. 15 MG 28*1 CPR.SS BLIST.	OTSUKA PHARMACEUTICAL NETH...	N05AX12	ARIPIPRAZOLE			
NO Legemiddelv...	ABILIFY	OTSUKA PHARMACEUTICAL NETH...	N05AX12	ARIPIPRAZOL	SMELTETA...	15 MG	28STK

Fig. 2: Examples of semi-structured and structured drug data.

In this paper, we describe the methods we use for parsing the data from individual data sets. The result of the work is the same data but fully structured. The parsing has to be performed repeatedly as the suppliers provide new drug data sets with updated prices and other drug information. The frequency of updates depends on national authorities and can vary from one day to one month.

1.1 Paper Outline

Section 2 mentions similar tasks in various domains, Section 3 describes the available data. In Section 4, we provide detailed information of the two basic methods we have used. Section 5 contains results of cross-validated evaluation. In Section 4, we plan further work on the topic.

2 Related Work

Tabular data are a very common form of information transfer. However, *tabular* not always means fully *structured*, i.e. usable by computer programs. A large survey on table understanding [2] describes various steps of table understanding in terms of dimension, nesting, generalization, and further processing (e.g., recognition of scanned tables) out of the scope of this work. Our case is one of the easiest: understanding of 1D nesting.

Understanding tables is a common task in the processing of Web documents since web tags for tables are also used for layout. [3] describes various challenges of table understanding in the context of the Web.

A major approach to table understanding is rule-based, as described e.g., in [4]. However, table cell parsing has a lot in common with sequence parsing and named entity recognition. We use two approaches, rule-based (regular expressions) and machine learning based.

3 Data Characteristics

In November 2019, Price Monitor database contained 115 million drug records that are daily growing. The data comes from more than 100 data sources and more than 45 countries in different languages. Among all the provided drug data, we are only interested in those that are usually put together in an unstructured form that makes it complicated to parse. Those data are:

- DRUG NAME (e.g., *Humira*, *Maxitrol*, *Nalgesin*)
- DOSAGE STRENGTH – indicates the amount of active substance in each dosage (e.g., *3 mg/ml*, *1.5 mg*, *100 ic/ml*)
- PACKAGE SIZE – indicates size of the package for certain drug (e.g., *28*, *2* $\times$ *330 ml*, *500 ml*)
- DOSAGE FORM – form of a drug product which contains active substance and other ingredients (e.g., solution for injection, tablet, concentrate for infusion, capsules, cream)

Because the suppliers provide this data in various forms, we divided the subset of data sets into three categories:

- YELLOW – all the data we are interested in are split into desired columns. The data are either split in the provided data sets so that no further processing is needed, or the data are split by simple processing in the form of regular expressions.
- BLUE – some of the data are split into the right columns, but some are not. For example, a name and a dosage strength are in separate columns, while a dosage form and a package size are in the same column.
- GRAY – all the data are in one column and often with different positions of values. For example, a brand name is not always at the beginning of a column, or a package size is not always followed by a dosage form in the same data set.

The data in the yellow category are entirely structured, which also means they are very uniform and contain precisely the estimated information. The yellow category contains 16 different data sets with 103 thousand records, which makes it the smallest one. The blue category contains 20 different data sets with 1.1 million records. The rest of the data sets are in the gray category. Because of the unstructured character of the data sets in the last two categories, columns contain other unnecessary information we could not easily eliminate.

3.1 Languages

The suppliers provide their data sets mostly in English, but some of them are only in their official language and therefore we need to tackle with the various cross-lingual variants (e.g., *tablet*, *tablety*, *tablett*, *tavoletta*) as well as non-Unicode characters. We also need to deal with various abbreviated words (e.g., *ml*, *mg*, *inj sol*, *tabl*, *tbl*, *filmtabl*).

4 Methods

4.1 Regular Expressions

We used regular expressions as the first method for parsing the data. However, we could use this method only for a small number of data sets and after a detailed analysis of data. We have to take into account positions of values and find patterns in their representation. Some data sets always respect their predefined format that allowed us, for instance, to rely on a brand name being at the beginning of line always followed by the same separator; or the same type of value always being in the parentheses. This method helps us mainly with enlarging the yellow category that we later used as an initial training set for another approach.

4.2 Conditional Random Fields

Conditional Random Fields (CRFs, [5]), a discriminative classifier, is a widely used statistical method for sequenced prediction. With the possibility to take context into account, it appears to be a promising method for parsing drug information.

Firstly, we had to create an appropriate training set. The first training set consists only from data sets from the yellow category, which contains already structured data, and therefore it was easy to label them. The training set was structured as a set of rows where every row represents one drug with its drug information. Every row was split into words and every word had its label. Because we are interested only about certain drug information, we used labels as DRUG NAME, DOSAGE STRENGTH, DOSAGE FORM and PACKAGE SIZE. For other unrelated drug information and punctuation marks, such as brackets, we used label OTHER. The data in training sets for the CRF method are also usually labeled with part-of-speech (POS) tags. Since our drug data are not in the form of sentences, we decided to omit this method because it would not have any additional value.

Secondly, we specified feature functions that are the key components of CRFs because they affect probabilities for sequence labeling. We started with functions such as word identity, word suffix and prefix, and whether the word is a number or punctuation. Later we experimented with more specific functions for our data, and we also specified the feature functions for neighboring words.

We set several experiments using `sklearn_crfsuite.CRF`[3] with various feature functions and training parameters described in detail in Section 4.3. For analyzing the feature weights of the model, we used `eli5`[4] Python library that provides transition features table and also state features table.

4.3 Experiments

We started with training data extracted only from the yellow category. The first experiments have shown weak predictions on data sets in languages that were not covered in the training set, especially in predictions of DOSAGE FORM (e.g., *tablet* and *tavoletta*). On the other hand, predictions of DRUG NAME or DOSAGE STRENGTH have shown above-average results since drug names usually have the same, or very similar, name in different languages, and DOSAGE STRENGTH usually follows a similar format (e.g., *X* ml/ *X* mg, *X* mg, *X* ml, *X* g).

Another finding was that because of the small amount of data in the training set, the predictions were biased. As we can see in Table 1, the model remembered whole words and assigned them high prediction weights. For instance, feature weights for Swedish words *peritonealdialysvätska* and *infusionsvätska* were relatively large, taking into account the fact that the values appeared only in one specific data set. As a result, the model could not perform well on new data on which it was not trained on.

To reduce over-fitting we used two different approaches, one is to enlarge the training set with more diverse data, the second is regularization.

Since we could not easily parse and label data sets from the blue category by regular expressions because of their inconsistent format, we labeled them with the classifier trained on the initial training set and then fix incorrect labels by hand. This approach allowed us to iteratively enrich the training set with data from the blue category and improved predicted results on a wider range of data with different drugs and in different languages. After several iterations, when the training set was large and diverse enough, we also labeled some data sets from the gray category to enrich the training set even more.

Another option to prevent over-fitting is to tune training parameters, especially regularization parameters L1 and L2. Regularization [1] is a smoothing technique that adds some penalty as the model complexity increases and the model consists of a large number of features. Because L1 regularization leads to feature selection and produces a sparse model by eliminating less important features [1], we tried to train a model using this technique. The cross-validation results have shown a significant change of weight from +6.780 to +4.390 for word *peritonealdialysvätska*; the model stopped to rely on particular words and started to use context more, which led to better generalization.

We performed further improvement in prediction by adding a feature that takes the prefix of a word into account. Many words in the domain start with the same prefix and differ in the endings in different languages. For example,

[3] `https://github.com/TeamHG-Memex/sklearn-crfsuite`

[4] `https://pypi.org/project/eli5/`

Table 1: Example of feature weights for dosage form after the first experiment

y='DOSAGE FORM'	top features
Weight	Feature
+10.693	word.lower():tablet
+7.471	-1:word.lower():surepal
+7.469	word.lower():tabletės
+7.270	word.lower():gelis
+6.921	-1:word.lower():trockensub
+6.780	+1:word.lower():peritonealdialysvätska
+6.513	+1:word.lower():infusionsvätska
+6.176	word[-3:]:eet
+6.061	-1:word.lower():stk
+5.872	word[-3:]:tfl
+5.808	word.lower():tabletė
+5.797	+1:word.lower():injektionsvätska
+5.555	word.lower():por
+5.555	word[-3:]:por
+5.501	word[-3:]:tti
+5.407	word.lower():capsule
+5.243	word.lower():krem
+5.221	word[-3:]:kum
	...
-9.010	word.isdigit()

the word *tablet*: in Czech it is *tableta*, in Swedish *tablett*, in German *tablette*, in shortened form can be *tabl*. After adding feature `'word[:-3]': word[:-3]` into the set of features, the prediction of DOSAGE FORM improved significantly for words with the same prefix.

Another feature – `features['BOS']`, which stands for *Beginning Of Sentence* – helps to predict the name of the drug as it is in most cases the first word in the sequence.

Since the drug data contains a lot of short words, such as *10 ml/mg*, *10 ml*, *1 vial of injection*, we decided to add features not only for 1, but also for 2 words before and after the current one to cover the context better.

We also noticed a small improvement by adding features such as `is_unit()` and `is_punctuation()` for words.

After those improvements, we achieved satisfactory results for the next iterations of the data labeling, and we could use the data from the blue category for making the training set more extensive and more diverse.

5 Results

The final training set consists of 1,687,187 drugs from all yellow, 4 blue, and 2 gray categories. The table shows the final 5-fold cross-validated results.

Table 2: 5-fold cross-validated results

	precision	recall	f1-score	support
DOSAGE FORM	0.95	0.94	0.94	388,489
DRUG NAME	0.94	0.92	0.93	257,298
OTHER	0.93	0.91	0.92	98,360
PACKAGE SIZE	0.94	0.94	0.94	391,810
DOSAGE STRENGTH	0.96	0.98	0.97	551,230
accuracy			0.95	1,687,187
macro avg	0.94	0.94	0.94	1,687,187
weighted avg	0.95	0.95	0.95	1,687,187

5.1 Error Analysis

After feature and training optimization and enlarging the training set with more various drug data, there are still some errors that often occur.

For instance, the model was trained that higher number values, such as *100*, *200* are more likely a part of DOSAGE STRENGTH, whereas smaller number values, such as *10*, *24*, or *50*, are more likely PACKAGE SIZE. However, when a bigger package of a drug appears (*100 tablets*), the model incorrectly predicts the value as DOSAGE STRENGTH.

Another error that occurs is related to the order of values to be predicted. The most common order is a sequence of brand name, dosage strength, package size, and dosage form.

Tables 4a and 4b illustrate an incorrect prediction when the values are provided in a less common order. We can see that prediction of DOSAGE STRENGTH value is missing and values related to DOSAGE STRENGTH and PACKAGE SIZE are predicted as OTHER.

As we did not use for training all the provided data from all categories, there are still errors in predicting values on unknown words and in languages that are not covered in the training set.

6 Conclusion and Future Work

In this work, we created a parser for records with pharmaceutical data. The purpose is to split the text into appropriate predefined columns: DRUG NAME, DOSAGE STRENGTH, DOSAGE FORM, PACKAGE SIZE, and OTHER. For roughly

Table 3: Example of correct and incorrect PACKAGE SIZE prediction.

DRUG NAME	DOSAGE STRENGTH	PACKAGE SIZE	DOSAGE FORM
viron	200 mg	70	kapsul
viron	200 mg 168		kapsul

Table 4a: Example of input data

	PACKAGE SIZE
medroxyprogesterone acetate 150 mg/ml inj,susp	1ml

Table 4b: Incorrectly labeled data

DRUG NAME	DOSAGE FORM	OTHER	PACKAGE SIZE
medroxyprogesterone acetate 150 /	inj, sus	mg ml	1 ml

one-quarter of the data, the splitting is not needed since the data already has the desired structure. We used these records as the training data and iteratively added new training examples by using regular expressions and conditional random fields. The present F1-measure in the 5-folded cross-validation evaluation is 0.95.

One future direction is to experiment more with the iterative approach, add new feature functions and discover inconsistencies in the training data.

Another future direction is to experiment with recurrent neural networks (RNNs) since in similar tasks such as named entity recognition, RNNs used together with CRFs provide the state-of-the-art results.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin infrastructure LM2015071 and OP VVV project CZ.02.1.01/0.0/0.0/16_013/0001781.

References

1. Chaturvedi, R., Arora, D., Singh, P.: Conditional random fields and regularization for efficient label prediction. ARPN Journal of Engineering and Applied Sciences **13**(20), 8332–8336 (Oct 2018)
2. Embley, D.W., Hurst, M., Lopresti, D., Nagy, G.: Table-processing paradigms: a research survey. International Journal of Document Analysis and Recognition (IJDAR) **8**(2), 66–86 (Jun 2006). https://doi.org/10.1007/s10032-006-0017-x, `https://doi.org/10.1007/s10032-006-0017-x`
3. Hurst, M.: Layout and language: Challenges for table understanding on the web. In: Proceedings of the International Workshop on Web Document Analysis. pp. 27–30 (2001)
4. Shigarov, A.: Rule-based table analysis and interpretation. In: Dregvaite, G., Damasevicius, R. (eds.) Information and Software Technologies. pp. 175–186. Springer International Publishing, Cham (2015)
5. Sutton, C., McCallum, A.: An introduction to conditional random fields. Foundations and Trends in Machine Learning **4**(4), 267–373 (2012)

Towards Universal Hyphenation Patterns

Petr Sojka and Ondřej Sojka

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
sojka@fi.muni.cz, ondrej.sojka@gmail.com

Abstract. Hyphenation is at the core of every document preparation system, being that typesetting system such as TEX or modern web browser. For every language, there have to be algorithms, rules, or patterns hyphenating according to that. We are proposing the development of generic hyphenation patterns for a set of languages sharing the same principles, e.g., for all syllable-based languages. We have tested this idea by the development of Czechoslovak hyphenation patterns. At the minimal price of a tiny increase in the size of hyphenation patterns, we have shown that further development of universal syllabic hyphenation patterns is feasible.

Keywords: hyphenation, hyphenation patterns, patgen, syllabification, Unicode, TEX, syllabic hyphenation, Czech, Slovak

"Any respectable word processing package includes a hyphenation facility. Those based on an algorithm, also called logic systems, often break words incorrectly."
Major Keary in [6]

1 Introduction

Hyphenation is at the core of every document preparation system, be it TEX or any modern web browser.[1] There are about 5,000 languages supported by Unicode Consortium that are still in use today. In a digital typography system supporting Unicode and its languages in full, there should be support: algorithms, rules, or language hyphenation patterns. Recently, there were attempts to tackle the word segmentation problem in different languages by Shao et al. [9], primarily for speech recognition or language representation tasks, where the algorithm is error-prone – small number of errors is tolerated. On the contrary, in a typesetting system like TEX, errors in hyphenation are not tolerated at all – all exceptions have to be covered by the algorithm.

Current typesetting support in the TEXlive distribution contains [8] hyphenation patterns for about 80 different languages. All these patterns have to be loaded into TEX's memory at the start of every compilation, which slows down compilation significantly.

There are essentially two quite different approaches to hyphenation:

[1] Cascading style sheet version 3 hyphenation implementation is supported in Firefox and Safari since 2011.

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 63–68, 2019.

etymology-based Rule is to cut word on the border of a compound word or the border of the stem and ending or prefix or negation. A typical example is British hyphenation rules created by the Oxford University Press [1].

phonology-based Hyphenation based on the pronunciation of syllables allows reading text with hyphenated lines similarly or the same as if the hyphenation were not there. This pragmatic approach is preferred by the American publishers [4] and the Chicago Manual of Style [2].

In this paper, we evaluate the feasibility of the development of universal phonology-based (syllabic) hyphenation patterns. We describe the development from word lists of Czech [11,15,12] and Slovak [14] used on the web pages. We describe the reproducible approach, and document the reproducible workflow and resources in the public repositories as a language resource and methods to be followed.

"Hyphenation does not lend itself to any set of unequivocal rules. Indeed, the many exceptions and disagreements suggest it is all something dreamed up at an anarchists' convention." Major Keary in [6]

2 Methods

The core idea is to develop common hyphenation patterns for phonology-based languages. In the case these languages share the pronunciation rules, homographs from different languages typically do not cause problems, as they are hyphenated the same. The rare cases that hyphenation is dictated by the seam of compound word contrary to phonology (`ro-zum` vs. `roz-um`) could be solved by not allowing the hyphenation.

Recently, we have shown that the approach to generate hyphenation patterns from word list by program `patgen` is unreasonably effective [16]. One can set the parameters of the generation process so that the patterns cover 100% of hyphenation points, and the size of the patterns remains reasonably small. For the Czech language, hyphenation points from 3,000,000 hyphenated words are squeezed into 30,000 bytes of patterns, stored as in the compressed trie data structure. That means achieving a compression ratio of several orders of magnitude with 100% coverage and nearly zero errors. [16] For a similar language such as Slovak, the pronunciation is very similar, syllable-forming principles are the same, and also compositional rules and prefixes are pretty close, if not identical.

We have decided to verify the approach by developing hyphenation patterns that will hyphenate both Czech and Slovak words without errors, with only a few missed hyphens. That means that only words like `oblít` will not be hyphenated, because the typesetting system cannot decide in which meaning the word is used: `o-blít` or `ob-lít`.

To generate these hyphenation patterns, we needed to create lists of correctly hyphenated Czech and Slovak words.

Data Preparation

For our work, word lists with frequencies for Czech and Slovak were donated from the TenTen family of corpora [5,7]. Only words that occurred more than ten times were used in further processing.

Czech word list was cleaned up and extended as described by us in [16], using Czech morphological analyzer `majka`. Final word list `cs-all-cstenten.wls` has 606,494 words.

For Slovak, we have got 1,048,860 Slovak words with frequency higher than 10 from SkTenTen corpora from 2011 [5]. Filtering only words containing ISO-Latin2 characters we obtained file `sk-all-latin2.wls` with 991,552 words.

Together, we have 1,319,334 Czech and Slovak words in `cssk-all-join.wls`, of which 139,356 were contained in both word lists: `cssk-all-intersect.wls`.

Pattern Development

The workflow of Czechoslovak pattern development is illustrated in Figure 1 on the following page. We have used recent accurate Czech patterns [16] for hyphenation of the joint Czech and Slovak word list. We had to manually fix bad hyphenation typically near the prefix and stem of words when phoneme-based hyphenation was one character close to the seam of the prefix or compound word: `neja-traktivnější`, `neja-teističtější`, `neje-kologičtější`.

We have then hyphenated words used in both languages also by current Slovak patterns. There were only a few word hyphenations that needed to be corrected – we created the `sk-corrections.wlh` that contained the fixed hyphenated words. Finally, we used them with a higher weight to generate final Czechoslovak hyphenated patterns.

Results

We have tried several parameters to generate Czechoslovak patterns, namely those developed in previous research [16], e.g. parameters for size- and correct-optimized set of patterns. After short fine tuning we have generated patterns that are both correct (Table 2 on page 67) and small (Table 1 on page 67). That means that `patgen` was able to generalize hyphenation rules common for both languages with a negligible enlargement of the generated patterns.

We have made all results and workflow reproducible by putting all files and necessary software (scripts, Makefiles) publicly available in our repository [10].

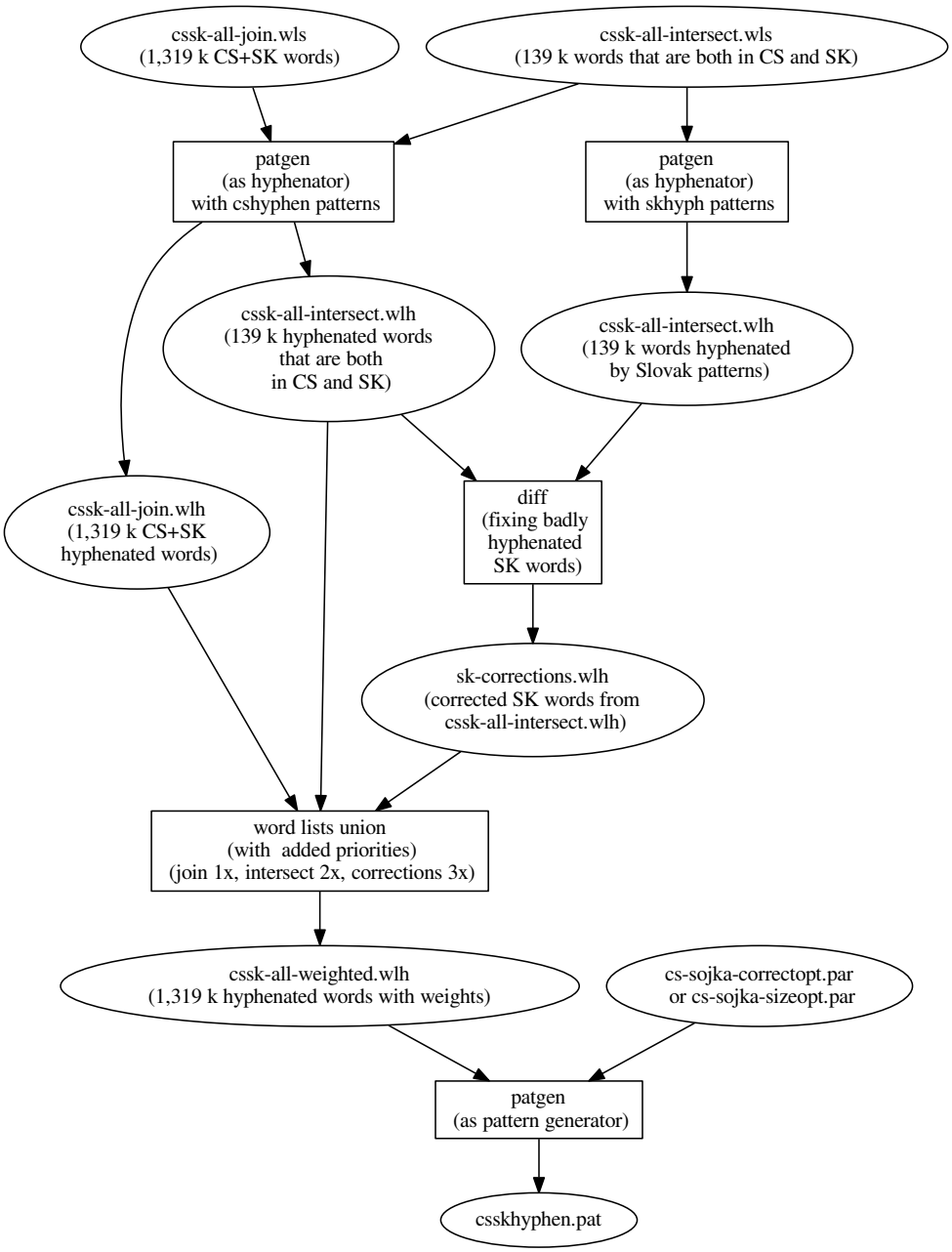


Fig. 1: The development process of the new Czechoslovak patterns. Bootstrapping with Czech patterns, checking and fixing with higher weight Slovak words that are common with Czech ones.

Table 1: Statistics from the generation of Czechoslovak hyphenation patterns with size optimized parameters.

Level	Patterns	Good	Bad	Missed	Lengths	Params
1	491	4,126,566	917,233	37,871	1 3	1 2 20
2	1,651	3,610,957	1,244	553,480	2 4	2 1 8
3	4,031	4,153,820	21,816	10,617	3 5	1 4 7
4	2,647	4,150,588	0	13,849	4 7	3 2 1

Table 2: Statistics from the generation of Czechoslovak hyphenation patterns with correct optimized parameters.

Level	Patterns	Good	Bad	Missed	Lengths	Params
1	2,221	4,082,523	345,325	81,914	1 3	1 5 1
2	2,237	4,071,117	9,416	93,320	1 3	1 5 1
3	4,337	4,164,049	11,001	395	2 6	1 3 1
4	2,647	4,162,859	0	1,625	2 7	1 3 1

Table 3: Comparison of the efficiency of different approaches to hyphenating Czech and Slovak. Note that the Czechoslovak patterns are comparable in size and quality to single-language ones – there is only a negligible difference compared e.g., to purely Czech patterns.

Word list	Patterns	Good	Bad	Missed	Size	# Patterns
Czechoslovak	sizeopt	99.67%	0.00%	0.33%	32 kB	5,679
Czechoslovak	correctopt	99.96%	0.00%	0.04%	48 kB	8,199
Czech	correctopt [16]	99.76%	2.94%	0.24%	30 kB	5,593
Czech	sizeopt [16]	98.95%	2.80%	1.05%	19 kB	3,816
Slovak	[13, Table 1, `patgen`]	99.94%	0.01%	0.06%	56 kB	2,347
Slovak	[3, by hand]	N/A	N/A	N/A	20 kB	2,467

> "Esoteric Nonsense? Hyphenation is neither anarchy nor the sole province of pedants and pedagogues.... If the author wants to attract and hold an audience, then hyphenation needs just as careful attention as any other aspect of presentation."
>
> Major Keary in [6]

3 Conclusion and Future Works

We have shown that the development of common hyphenation patterns for languages with similar pronunciation is feasible. The resulting Czechoslovak patterns are only slightly bigger than single-language patterns and hyphenate the source word list without a single error.

Development is expected to continue in the repository [10]. Final word lists and versions of hyphenation patterns will be deposited into the LINDAT-Clarin archive.

We will double-check the hyphenated word lists with members of Czechoslovak TEX Users Group CSTUG. We will finally offer the new patterns for "Czechoslovak language" to the TEXlive distribution, creating the first language support package to be shared by multiple languages.

Acknowledgement This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin infrastructure LM2015071. We are indebted to Don Knuth for questioning common properties of Czech and Slovak hyphenation during our presentation of [16] at TUG 2019 that lead us into this direction.

References

1. Allen, R.: The Oxford Spelling Dictionary, The Oxford Library of English Usage, vol. II. Oxford University Press (1990)
2. Anonymous: The Chicago Manual of Style. University of Chicago Press, Chicago, 17 edn. (Sep 2017)
3. Chlebíková, J.: Ako rozdeliť (slovo) Československo (How to Hyphenate (word) Czechoslovakia). CSTUG Bulletin **1**(4), 10–13 (Apr 1991)
4. Gove, P.B., Webster, M.: Webster's Third New International Dictionary of the English language Unabridged. Merriam-Webster Inc., Springfield, Massachusetts, U.S.A (Jan 2002)
5. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., Suchomel, V.: The TenTen Corpus Family. In: Proc. of 7th International Corpus Linguistics Conference (CL). pp. 125–127. Lancaster (Jul 2013)
6. Keary, M.: On hyphenation – anarchy of pedantry. PC Update, The magazine of the Melbourne PC User Group (2005), https://web.archive.org/web/20050310054738/http://www.melbpc.org.au/pcupdate/9100/9112article4.htm
7. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. In: Proceedings of the Eleventh EURALEX International Congress. pp. 105–116. Lorient, France (2004)
8. Reutenauer, A., Miklavec, M.: TEX hyphenation patterns, https://tug.org/texhyphen/, accessed 2019-11-24
9. Shao, Y., Hardmeier, C., Nivre, J.: Universal Word Segmentation: Implementation and Interpretation. Transactions of the Association for Computational Linguistics **6**, 421–435 (2018). https://doi.org/10.1162/tacl_a_00033, https://www.aclweb.org/anthology/Q18-1033
10. Sojka, O., Sojka, P.: cshyphen repository, https://github.com/tensojka/cshyphen
11. Sojka, P.: Notes on Compound Word Hyphenation in TEX. TUGboat **16**(3), 290–297 (1995)
12. Sojka, P.: Hyphenation on Demand. TUGboat **20**(3), 241–247 (1999)
13. Sojka, P.: Slovenské vzory dělení: čas pro změnu? In: Proceedings of SLT 2004, 4th seminar on Linux and TEX. pp. 67–72. Konvoj, Znojmo (2004)
14. Sojka, P.: Slovenské vzory dělení: čas pro změnu? (Slovak Hyphenation Patterns: A Time for Change?). CSTUG Bulletin **14**(3–4), 183–189 (2004)
15. Sojka, P., Ševeček, P.: Hyphenation in TEX – Quo Vadis? TUGboat **16**(3), 280–289 (1995)
16. Sojka, P., Sojka, O.: The unreasonable effectiveness of pattern generation. TUGboat **40**(2), 187–193 (2019), https://tug.org/TUGboat/tb40-2/tb125sojka-patgen.pdf

Part III

Semantics and Language Modelling

Adjustment of Goal-driven Resolution for Natural Language Processing in TIL

Marie Duží, Michal Fait, and Marek Menšík

VSB-Technical University of Ostrava, Department of Computer Science FEI,
17. listopadu 15, 708 33 Ostrava, Czech Republic
marie.duzi@vsb.cz, michal.fait@vsb.cz, marek.mensik@vsb.cz

Abstract. The paper deals with natural language reasoning and question answering. Having a fine-grained analysis of natural language sentences in the form of TIL (Transparent Intensional Logic) constructions, we apply the General Resolution Method (GRM) with its goal-driven strategy to answer the question (goal) raised on the natural language data. Not only that, we want to answer in an 'intelligent' way, so that to provide logical consequences entailed by the data. From this point of view, GRM appears to be one of the most plausible proof techniques. There are two main new results presented here. First, we found out that it is not always possible to apply all the necessary adjustments of the input constructions first, and then to go on in a standard way by applying the algorithm of the transformation of propositional constructions into the Skolem clausal form followed by the GRM goal-driven resolution techniques. There are plenty of features special for the rich natural language semantics that are dealt with by TIL technical rules and these rules must be integrated with the process of the goal-driven resolution technique rather than separated from it. Second, the strategy of generating resolvents from a given knowledge base cannot be strictly goal-driven. Though we start with a given goal/question, it may happen that there is a point at which we have to make a step aside. We have to apply those special TIL technical rules on another clause first, and only then it is possible to go on with the process of resolving clauses with a given goal. Otherwise our inference machine would be heavily under-inferring, which is not desirable, of course. We demonstrate these new results by two simple examples. The first one deals with property modifiers and anaphoric references. Anaphoric references are dealt with by our substitution method, and the second example demonstrates reasoning with factive verbs like 'knowing' together with definite descriptions and anaphoric references again. Since the definite description occurs *de re* here, we substitute a pointer to the individual referred to for the respective anaphoric pronoun.

Keywords: natural language reasoning, Transparent Intensional Logic, TIL, General Resolution Method, goal-driven strategy, property modifiers, factive verbs, anaphoric references

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 71–81, 2019.

1 Introduction

Natural language processing, computational linguistics and logic are the disciplines that have more in common than it might seem at first sight. The hard work of many linguists supported by computers produces large corpora of analyzed text data where a lot of information is contained. Logicians contribute by their rational arguments to logically organize and analyze the data so that we might "teach the computers to be intelligent". Artificial intelligence is flourishing. Or not? Actually, it turns out that there is an information overload. People are not able to know their way around the information labyrinth; internet is infested with fake news; artificial "intelligence" is not intelligent.

At the effort of improving the situation, we started the project on question answering system over natural language texts [5]. In this project linguists and logicians work hand-in-hand. After linguistic preprocessing of the texts a corpus of fine-grained logically structured semantic data has been produced. The project goal is this. Given a question the system should be able to answer the question in a more intelligent way than by just providing explicitly recorded text data sought by keywords. To this end we are building up an inference machine that operates on those logical structures so that not only to provide explicit textual knowledge but also to compute inferable logical knowledge [7] such that rational human agents would produce, if only this were not beyond their time and space capacity.

Our background logic is the well-known system of Transparent Intensional Logic (TIL) with its *procedural* rather than set-theoretic semantics. Hence, meaning of a sentence is an abstract procedure encoded by the sentence that can be viewed as an instruction how, in any possible world and time, to evaluate the truth-value of the sentence, if any. These procedures are known as TIL *constructions*. There are six kinds of such constructions defined, namely *variables*, *Trivialization*, *Composition*, *(λ)-Closure*, *Execution* and *Double Execution*. While the first two are atomic constructions, the latter four are molecular. Atomic constructions supply objects on which molecular constructions operate; where X is an object what so ever of TIL ontology, Trivialization ${}^0X$ produces X. *Composition* $[FA_1 \dots A_m]$ is the procedure of applying a function produced by F to its arguments produced by $A_1, \dots, A_m$ to obtain the value of the function; dually, Closure $[\lambda x_1 \dots x_m\, C]$ is the procedure of declaring or constructing a function by abstracting over the values of λ-bound variables. As obvious, TIL is a typed λ-calculus that operates on functions (intensional level) and their values (extensional level), as ordinary λ-calculi do; in addition to this dichotomy, there is however the highest *hyperintensional* level of procedures producing lower-level objects. And since these procedures can serve as objects on which other higher-order procedures operate, we can execute procedures twice over. To this end there is the construction called *Double Execution*. To avoid vicious circle problem and keep track in the rich hierarchy of logical strata, TIL ontology is organized into *ramified hierarchy of types* built over a base. For the purpose of natural language processing we use the *epistemic base* consisting of for atomic types, namely o (the set of truth-values), ι (individuals), τ (times or real numbers)

and ω (possible worlds). The type of constructions is $*_n$ where n is the order of a construction.

Empirical sentences and terms denote (PWS-)*intensions*, objects of types $((\alpha\tau)\omega)$, or $\alpha_{\tau\omega}$, for short. Where variables w, t range over possible worlds $(w \to \omega)$ and times $(t \to \tau)$, respectively, constructions of intensions are usually Closures of the form $\lambda w \lambda t\,[\ldots w \ldots t \ldots]$. For a simple example, where *Surgeon* is a property of individuals of type $(o\iota)_{\tau\omega}$ and *John* is an individual of type ι, the sentence "John is a surgeon" encodes as its meaning the hyper-proposition

$$\lambda w \lambda t\,[[[^0Surgeon\ w]\ t]\ {}^0John], \text{ or } \lambda w \lambda t\,[{}^0Surgeon_{wt}\ {}^0John], \text{ for short.}$$

Hence, the input information base on which our inference machine operates, is a large collection of such hyper-propositions over which we ask queries.

For instance, given questions like "Is there a surgeon?", "Who is it?" the system derives answers logically entailed by the base like "yes", "he is John". This is very simple, of course. A given question is a *goal* the answer to which the system derives from the knowledge base. Thus a proof-calculus with a *goal-driven strategy* and backward chaining proof method [11, Ch.9] such as the well-known *General Resolution Method* (GRM) seems to be a natural choice here and the technique of *Resolution Theorem Proving* is broadly applied in artificial intelligence.

In [6] we briefly demonstrated application of GRM by the 'Sport Club' example. Since the FOL (first-order predicate logic) general resolution method operates on formulas in their clausal form, we specified the algorithm of transferring hyper-propositions, i.e. closed constructions that are typed to construct propositions, into the clausal form. Yet natural language is semantically much richer than the language of FOL. There are numerous semantic features of natural language that do not appear in a formal, unambiguous logical language. Apart from the problem of high ambiguity, processing natural language must deal with propositional and notional *attitudes*, *property modifiers*, *anaphoric references*, *modalities*, different grammatical *tenses* and *time references*, *definite descriptions* and presuppositions connected with them, other *presupposition triggers* like topic-focus articulation within a sentence, etc etc.

TIL is the system where such semantically salient special features are logically tractable. We have got special technical rules and methods to operate *in* a hyperintensional context, to deal with *de dicto* vs. *de re* attitudes, presupposition triggers, rules for factive attitudes like *knowing* or *regretting*, *partiality* (sentences with truth-value gaps), grammatical tenses and reference times, and so like. These technicalities include, inter alia, the *substitution method*, i.e. application of the special functions that operate on constructions, namely $Sub/(*_n\ *_n\ *_n *_n)$ and $Tr/(*\,\alpha)$ together with Double Execution, properties of propositions like *True*, *False* and *Undef*, transition from a property modifier to a property (pseudo-detachment, i.e. the *left subsectivity* rule [10]), and many others. Compared to this semantic richness, the semantics of FOL formulas in their clausal form is much simpler. Hence, there is a problem how to apply a formal FOL method such as general resolution without losing semantic information encoded in TIL

constructions. In previous works ([1], [6]) we assumed that it would be possible to pre-process TIL constructions into the form plausible for the application of the algorithm of transformation into the Skolem clausal form first, and then apply the algorithm so that goal-driven resolution can start.

However, as it turns out, this way is under-inferring. It can be the case that we might derive the respective answer entailed by the knowledge base if only we could harmonically integrate those special TIL rules with the goal-driven resolution process. The goal of this paper and its novel contribution is a proposal of such a method. Using examples, we demonstrate how to deal with property modifiers, anaphoric references, factive propositional attitudes and definite descriptions.

The rest of the paper is organized as follows. Reasoning with property modifiers and anaphoric references is demonstrated by an example involving a 'married man' in Section 2. Section 3 deals with factive propositional attitudes, definite descriptions and anaphoric references. Concluding remarks can be found in Section 4.

2 Reasoning with property modifiers

Scenario. John is a married man. John's partner is Eve. Everybody who is married believes, that his/her partner is amazing.
Question. Does John believe that Eve is amazing?

Formalization starts with assigning *types* to the objects that receive mention in our scenario:
$John, Eve/\iota$; $Married^m/((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$ a property modifier; $Married/(o\iota)_{\tau\omega}$ the property of being married; $Amazing/(o\iota)_{\tau\omega}$; $Partner/(\iota)_{\tau\omega}$; $Believe/(o\iota*)_{\tau\omega}$; $w \to \omega$; $t \to \tau$; $x, y \to \iota$.

The analysis of the three premises and the question comes down to these constructions:

A: $\lambda w\lambda t\,[[^0Married^m\ ^0Man]_{wt}\ ^0John]$
B: $\lambda w\lambda t\,[[^0Partner_{wt}\ ^0John] = {}^0Eve]$
C: $\lambda w\lambda t\,\forall x\,[[^0Married_{wt}\ x] \supset [^0Believe_{wt}\ x\ [^0Sub\ [^0Tr\ [^0Partner_{wt}\ x]]\ ^0y\ {}^0[\lambda w\lambda t\,[^0Amazing_{wt}\ y]]]]]$
Q: $\lambda w\lambda t\,[^0Believe_{wt}\ ^0John\ ^0[\lambda w\lambda t\,[^0Amazing_{wt}\ ^0Eve]]]$

Note that at this point we cannot adjust the premise (C), i.e. to evaluate the substitution, because this is a general rule. In other words, we do not know as yet which individuals should be substituted for x, and thus also for y.

The algorithm of transferring constructions into their Skolem clausal form [6] starts with the elimination of the left-most $\lambda w\lambda t$, and negation of the question (Q), thus obtaining the first goal (G):

1. Elimination of the left-most $\lambda w \lambda t$, and obtaining the first goal G by negating the question Q.
   A: $[[{}^0Married^m\ {}^0Man]_{wt}\ {}^0John]$
   B: $[[{}^0Partner_{wt}\ {}^0John] = {}^0Eve]$
   C: $\forall x\,[[{}^0Married_{wt}\ x] \supset [{}^0Believe_{wt}\ x\ [{}^0Sub\ [{}^0Tr\ [{}^0Partner_{wt}\ x]]\ {}^0y\ {}^0[\lambda w \lambda t\ [{}^0Amazing_{wt}\ y]]]]]$
   G: $\neg[{}^0Believe_{wt}\ {}^0John\ {}^0[\lambda w \lambda t\ [{}^0Amazing_{wt}\ {}^0Eve]]]$

2. Elimination of $\supset$
   A: $[[{}^0Married^m\ {}^0Man]_{wr}\ {}^0John]$
   B: $[[{}^0Partner_{wt}\ {}^0John] = {}^0Eve]$
   C: $\forall x\,[\neg[{}^0Married_{wt}\ x] \vee [{}^0Believe_{wt}\ x\ [{}^0Sub\ [{}^0Tr\ [{}^0Partner_{wt}\ x]]\ {}^0y\ {}^0[\lambda w \lambda t\ [{}^0Amazing_{wt}\ y]]]]]$
   G: $\neg[{}^0Believe_{wt}\ {}^0John\ {}^0[\lambda w \lambda t\ [{}^0Amazing_{wt}\ {}^0Eve]]]$

3. Elimination of $\forall$
   A: $[[{}^0Married^m\ {}^0Man]_{wt}\ {}^0John]$
   B: $[[{}^0Partner_{wt}\ {}^0John] = {}^0Eve]$
   C: $\neg[{}^0Married_{wt}\ x] \vee [{}^0Believe_{wt}\ x\ [{}^0Sub\ [{}^0Tr\ [{}^0Partner_{wt}\ x]]\ {}^0y\ {}^0[\lambda w \lambda t\ [{}^0Amazing_{wt}\ y]]]]$
   G: $\neg[{}^0Believe_{wt}\ {}^0John\ {}^0[\lambda w \lambda t\ [{}^0Amazing_{wt}\ {}^0Eve]]]$

   Basically, our constructions are in the Skolem clausal form now, though a little bit more complex form. The resolution process should start with the goal G and look for a clause where a positive constituent $[{}^0Believe_{wt}\ x\ ...]$ occurs. Obviously, it is the clause C. Resolution rule in FOL makes use of Robinson's *unification* algorithm. In principle, this algorithm substitutes terms for variables, which transforms in TIL into the substitution of constituents for variables. Yet, to unify constructions of the arguments of the function produced by ${}^0Believe_{wt}$ as they occur in the clauses G and C, it is not sufficient to substitute the constituent ${}^0John$ for the variable x. In addition, we have to exploit the clause B and substitute ${}^0Eve$ for $[{}^0Partner_{wt}\ {}^0John]$. As a result, we obtain

4. Unification and Resolution

$$[{}^0Believe_{wt}\ {}^0John\ [{}^0Sub\ [{}^0Tr\ {}^0Eve]\ {}^0y\ {}^0[\lambda w \lambda t\ [{}^0Amazing_{wt}\ y]]]]$$

   Next, the application of the functions *Sub* and *Tr* must be evaluated by applying these transformations:

$$[{}^0Tr\ {}^0Eve] \implies {}^0Eve$$

$$[{}^0Sub\ [{}^0Tr\ {}^0Eve]\ {}^0y\ {}^0[\lambda w \lambda t\ [{}^0Amazing_{wt}\ y]]] \implies {}^0[\lambda w \lambda t\ [{}^0Amazing_{wt}\ {}^0Eve]]$$

   As a result, we obtain the adjusted clause

   C': $\neg[{}^0Married_{wt}\ {}^0John] \vee [{}^0Believe_{wt}\ {}^0John\ {}^0[\lambda w \lambda t\ [{}^0Amazing_{wt}\ {}^0Eve]]]$

Only now can the clauses G and C′ be resolved so that the new goal is obtained:

R1: $\neg[{}^0Married_{wt}\ {}^0John]$ (G+C′)

However, this goal cannot be met, unless the rule of *left subsectivity* ([10], [9], [2]) that is universally valid for any kind of a property modifier is applied. Where $P^m \rightarrow ((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$ is a construction of a property modifier and $P \rightarrow (o\iota)_{\tau\omega}$ a construction of the property corresponding to the modifier, the rule is this:[1]

$$[[P^m Q]_{wt}\ x] \vdash [P_{wt}\ x]$$

In our case the rule results in $[[{}^0Married^m\ {}^0Man]_{wt}\ x] \vdash [{}^0Married_{wt}\ x]$. For the purpose of resolution method, we rewrite the rule into the implicative, hence clausal form, thus obtaining another clause

M: $\neg[[{}^0Married^m\ {}^0Man]_{wt}\ z] \vee [{}^0Married_{wt}\ z]$

Now the last goal R1 is easily met:

R2: $\neg[[{}^0Married^m\ {}^0Man]_{wt}\ {}^0John]$ (R1+M), ${}^0John/z$
R3: # (R2+A)

By applying an indirect proof we obtained the empty clause that cannot be satisfied, hence the answer to the question Q is YES.

By this simple example we demonstrated that it is not possible to evaluate constructions stemming from the special TIL techniques like the anaphoric substitution by means of the functions *Sub* and *Tr* in the phase of pre-processing constructions and their transformation into the Skolem clausal form. Rather, we have to integrate these techniques with the unification of clauses into the process of deriving resolvents. In addition, we also have to apply special rules that are rooted in the rich semantics of natural language like the rule of left subsectivity. We propose to specify such rules in the form of additional semantic clauses that are recorded in agent's ontology.

3 Reasoning with factive propositional attitudes

In this section we are going to demonstrate by an example reasoning with factive propositional attitudes like 'knowing that' for which special rules rendering the fact that the truth of the known sub-proposition is a presupposition of the whole proposition. In other words, if a proposition P is not true, then P can be neither

[1] The rigorous definition of the property corresponding to the respective modifier can be found in [2]; roughly, P is defined as the property of x such that there is a property q with respect to which x is a $[P^m q]$. For instance, a skillful surgeon is skillful as a surgeon. Hence, there is a property with respect to which a skillful surgeon is skillful.

known nor not known. Thus, the rules that we have to apply in this example are specified as follows.

$$[^0Know_{wt}\, a\, C] \vdash [^0True_{wt}\, ^2C]$$
$$\neg[^0Know_{wt}\, a\, C] \vdash [^0True_{wt}\, ^2C]$$

Types. $Know \rightarrow (o\iota *_n)_{\tau\omega}$; $a \rightarrow \iota$; $C \rightarrow *_n$; $^2C \rightarrow o_{\tau\omega}$; $True/(oo_{\tau\omega})_{\tau\omega}$.

In addition to these rules, we often need to apply the rule of *True elimination*:

$$[^0True_{wt}\, p] \vdash p_{wt}$$

Similarly as in the previous example, the above rules will be specified in their implicative form so that we obtain three additional clauses. In the resolution process we also make technical adjustments, in particular by applying the rule of ^{20}*-conversion*:

$$^{20}C = C$$

for any closed construction C that is typed to v-construct a non-procedural object of a type of order 1.

Scenario. The Mayor of Ostrava knows that the President of TUO does not know (yet) that he (the President) will go to Brussels. The President of TUO is prof. Snasel.
Question. Will prof. Snasel go to Brussels?

Formalization.

As always, first types: *Snasel*, *Brussels*$/\iota$; *Know*$/(o\iota *_n)_{\tau\omega}$; *President*(-of TUO), *Mayor*(-of Ostrava)$/\iota_{\tau\omega}$; *Go*$/(o\iota\iota)_{\tau\omega}$.

Premises:
A: $\lambda w\lambda t\, [^0Know_{wt}\, ^0Mayor_{wt}\, ^0[\lambda w\lambda t\, \neg[^0Know_{wt}\, ^0President_{wt}$
$[^0Sub\, [^0Tr\, ^0President_{wt}]\, ^0he\, ^0[\lambda w\lambda t\, [^0Go_{wt}\, he\, ^0Brussels]]]]]]$
B: $\lambda w\lambda t\, [^0President_{wt} = {}^0Snasel]$

Conclusion/question:
Q: $\lambda w\lambda t\, [^0Go_{wt}\, ^0Snasel\, ^0Brussels]$

In addition to these premises and conclusion, we have the above rules which result into three clauses M1, M2 and T:

M1: $[^0Know_{wt}\, x\, c] \supset [^0True_{wt}\, ^2c]$
M2: $\neg[^0Know_{wt}\, x\, c] \supset [^0True_{wt}\, ^2c]$
T: $[^0True_{wt}\, p] \supset p_{wt}$

Additional types. $c \rightarrow *_n$; $^2c \rightarrow o_{\tau\omega}$; $p \rightarrow o_{\tau\omega}$

The algorithm of transferring these constructions into their clausal form proceeds as follows:

1. Elimination of the left-most $\lambda w \lambda t$, renaming variables, negation of the question Q that results in the goal G.
   A: $[{}^0Know_{wt}\ {}^0Mayor_{wt}\ {}^0[\lambda w \lambda t\ \neg[{}^0Know_{wt}\ {}^0President_{wt}\ [{}^0Sub\ [{}^0Tr\ {}^0President_{wt}]\ {}^0he\ {}^0[\lambda w \lambda t\ [{}^0Go_{wt}\ he\ {}^0Brussels]]]]]]$
   B: $[{}^0President_{wt} = {}^0Snasel]$
   M1: $[{}^0Know_{wt}\ x\ c] \supset [{}^0True_{wt}\ {}^2c]$
   M2: $\neg[{}^0Know_{wt}\ y\ d] \supset [{}^0True_{wt}\ {}^2d]$
   T: $[{}^0True_{wt}\ p] \supset p_{wt}$
   G: $\neg[{}^0Go_{wt}\ {}^0Snasel\ {}^0Brussels]$

2. Elimination of $\supset$
   A: $[{}^0Know_{wt}\ {}^0Mayor_{wt}\ {}^0[\lambda w \lambda t\ \neg[{}^0Know_{wt}\ {}^0President_{wt}\ [{}^0Sub\ [{}^0Tr\ {}^0President_{wt}]\ {}^0he\ {}^0[\lambda w \lambda t\ [{}^0Go_{wt}\ he\ {}^0Brussels]]]]]]$
   B: $[{}^0President_{wt} = {}^0Snasel]$
   M1: $\neg[{}^0Know_{wt}\ x\ c] \vee [{}^0True_{wt}\ {}^2c]$
   M2: $[{}^0Know_{wt}\ y\ d] \vee [{}^0True_{wt}\ {}^2d]$
   T: $\neg[{}^0True_{wt}\ p] \vee p_{wt}$
   G: $\neg[{}^0Go_{wt}\ {}^0Snasel\ {}^0Brussels]$

   Our constructions (hyperpropositions) are in the Skolem clausal form now, and the process of resolution together with unification can start. Since the strategy is goal-driven, we aim at choosing a clause with a positive constituent $[{}^0Go_{wt}...]$. The only candidate is the clause A. However, there is a problem here. The constituent ${}^0Go$ occurs in the goal G extensionally, while the same constituent occurs in A in the hyperintensional context, i.e. closed by Trivialization and thus not amenable to logical operations.[2] Yet, the 'magic trick' of this argument consists in the fact that *Knowing* is a factivum. In other words, by applying the rules M1, M2 and T we can decrease the context down to the extensional level. This in turn means that before exploiting the goal G we have to make a 'step aside'. Before resolving the goal G with any other clause, we must resolve A, M1, M2 and T until the constituent ${}^0Go$ gets down to the extensional level. Here is how.

3. Unification and resolution
   R1: $[{}^0True_{wt}\ {}^{20}[\lambda w \lambda t\ \neg[{}^0Know_{wt}\ {}^0President_{wt}\ [{}^0Sub\ [{}^0Tr\ {}^0President_{wt}]\ {}^0he\ {}^0[\lambda w \lambda t\ [{}^0Go_{wt}\ he\ {}^0Brussels]]]]]]$
   (A+M1)
   ${}^0Mayor_{wt}/x,\ {}^0[\lambda w \lambda t\ \neg[{}^0Know_{wt}\ {}^0President_{wt}\ [{}^0Sub\ [{}^0Tr\ {}^0President_{wt}]\ {}^0he\ {}^0[\lambda w \lambda t\ [{}^0Go_{wt}\ he\ {}^0Brussels]]]]]/c$

[2] For details on the three kinds of context in which a construction can occur within another construction, see [4], and the details on hyperintensionally closed constructions can be found in [3] and [8].

R2: $^{20}[\lambda w\lambda t\ \neg[^0Know_{wt}\ ^0President_{wt}$
$[^0Sub\ [^0Tr\ ^0President_{wt}]\ ^0he^0[\lambda w\lambda t\ [^0Go_{wt}\ he\ ^0Brussels]]]]]_{wt}$

(R1+T)
$^{20}[\lambda w\lambda t\ \neg[^0Know_{wt}\ ^0President_{wt}$
$[^0Sub\ [^0Tr\ ^0President_{wt}]\ ^0he^0[\lambda w\lambda t\ [^0Go_{wt}\ he\ ^0Brussels]]]]]/p$

R3: $[^0True_{wt}\ ^2[^0Sub\ [^0Tr\ ^0President_{wt}]\ ^0he^0[\lambda w\lambda t\ [^0Go_{wt}\ he\ ^0Brussels]]]]$

(R2+M2)
^{20}-conversion,
restricted β-conversion,
$^0President_{wt}/y$,
$[^0Sub\ [^0Tr\ ^0President_{wt}]\ ^0he^0[\lambda w\lambda t\ [^0Go_{wt}\ he\ ^0Brussels]]]/d$

R4: $^2[^0Sub\ [^0Tr\ ^0President_{wt}]\ ^0he^0[\lambda w\lambda t\ [^0Go_{wt}\ he\ ^0Brussels]]]_{wt}$

(R3+T)
$^2[^0Sub\ [^0Tr\ ^0President_{wt}]\ ^0he^0[\lambda w\lambda t[^0Go_{wt}\ he\ ^0Brussels]]]/p$

To evaluate the substitution, i.e. to obtain the proposition v-constructed by R4, we make use of the clause B in order to substitute ^0Snasel for $^0President_{wt}$. Thus, we have

$^2[^0Sub\ [^0Tr\ ^0President_{wt}]\ ^0he^0[\lambda w\lambda t\ [^0Go_{wt}\ he\ ^0Brussels]]]_{wt} =$	R4, B
$^2[^0Sub\ [^0Tr\ ^0Snasel]\ ^0he^0[\lambda w\lambda t\ [^0Go_{wt}\ he\ ^0Brussels]]]_{wt} =$	*Sub*, *Tr*
$^{20}[\lambda w\lambda t\ [^0Go_{wt}\ ^0Snasel\ ^0Brussels]]_{wt} =$	^{20}-conversion, β
$[^0Go_{wt}\ ^0Snasel\ ^0Brussels]$	

As a result, we obtained an adjusted clause
R4': $[^0Go_{wt}\ ^0Snasel\ ^0Brussels]$
R5: # (R4'+G)

Hence, the answer to the question Q is YES.

By this example we demonstrated that though our strategy is goal-driven, we cannot proceed strictly from a goal to meet another goal. At the beginning of the resolution process, we used the goal G to determine the clause A as the one that might be suitable for meeting the goal G. However, to do so, we first had to resolve the clause A with other clauses (M1, M2, T), then adjust the result by means of the clause B and the special TIL technical rules, and only then could we resolve the result with our goal G to obtain an empty clause, and thus answer the question Q in positive.

Summarising, an adjustment of the general resolution method for natural language processing in TIL *cannot be strictly goal-driven*, which is another novel result of this paper.

4 Conclusion

In the paper we introduced reasoning with fine-grained semantics of natural language in the question-answer system based on Transparent Intensional Logic. We examined application of the General Resolution Method with its goal-driven strategy that is also characterized as backward chaining, because a given goal determines which clauses are selected and used for generating resolvents. Backward chaining starts with a goal (question or hypothesis) and works backwards from the consequent to the antecedent to see if any clause supports any of these consequents. We solved two problems. First, how to integrate special rules rooted in the rich natural language semantics with the process of generating resolvents. Second, we found out that due to these special rules the process cannot be strictly goal-driven; it starts with a given goal/question, yet it may happen that we have to make a 'step aside' in order to adjust other clauses first, and only then we can resolve.

Future research will be oriented to forward-chaining inference, as it is applied in the Gentzen natural deduction system. In such a system there is not a problem of smoothly integrating other rules as needed; rather, we will have to solve the problem how to answer a given question and not to get lost in the huge labyrinth of input data. Last but not least, we would eventually like to make a comparison of such two different approaches to the design of an inference machine for natural language processing using TIL.

Acknowledgements. This research was funded by the Grant Agency of the Czech Republic (GACR) project GA18-23891S "Hyperintensional Reasoning over Natural Language Text" and by the internal grant agency of VSB-Technical University of Ostrava, project No. SP2019/40, "Application of Formal Methods in Knowledge Modelling and Software Engineering I". Michal Fait was also supported by the Moravian-Silesian regional program No. RRC/10/2017 "Support of science and research in Moravian-Silesian region 2017" and by the EU project "Science without borders" No. CZ.02.2.69/0.0/0.0/16_027/0008463.

References

1. Číhalová, M., Duží, M., Ciprich, N., Menšík, M. (2010), 'Agents' reasoning using TIL-Script and Prolog', *Frontiers in Artificial Intelligence and Applications* 206: 135-154.
2. Duží, M. (2017): Property modifiers and intensional essentialism. *Computación y Sistemas*, vol. 21, No. 4, 2017, pp. 601–613. DOI: 10.13053/CyS-21-4-2811.
3. Duží, M. (2019): If structured propositions are logical procedures then how are procedures individuated? *Synthese* special issue on the Unity of propositions, vol. 196, No. 4, pp. 1249-1283. DOI: 10.1007/s11229-017-1595-5
4. Duží, M., Fait, M., Menšík, M. (2017): Context Recognition for a Hyperintensional Inference Machine. In the AIP proceeding of *ICNAAM 2016*, International Conference of Numerical Analysis and Applied Mathematics, vol. 1863, Article No. 330004
5. Duží, M., Horák, A. (2015): TIL as hyperintensional logic for natural language analysis. In *RASLAN 2015*, A. Horák, P. Rychlý and A. Rambousek (eds.), pp. 113-124.

6. Duží, M., Horák, A. (2019): Hyperintensional Reasoning based on Natural Language Analysis. To appear in the *International Journal of Uncertainty, Fuzziness & Knowledge-Based Systems (IJUFKS)*.
7. Duží, M., Menšík, M. (2017): Logic of Inferable Knowledge. In Jaakkola, H., Thalheim, B., Kiyoki, Y. and Yoshida, N., eds., Information Modelling and Knowledge Bases XXVIII, *Frontiers in Artificial Intelligence and Applications*, Amsterdam: IOS Press, vol. 292, 2017, pp. 405-425.
8. Fait, M., Duží, M. (2019): Substitution rules with respect to a context. In *Lecture Notes in Electrical Engineering* 554, I. Zelinka, P. Brandstetter, T.T. Dao, Vo. H. Duy, S.B. Kim eds., AETA 2018 – Recent Advances in Electrical Engineering and Related Sciences: Theory and Applications, Springer, pp. 55-66.
9. Jespersen, B. (2015): Structured lexical concepts, property modifiers, and Transparent Intensional Logic. *Philosophical Studies*, vol. 172, No. 2, pp. 321-345.
10. Jespersen, B. (2016): Left Subsectivity: How to Infer that a Round Peg is Round. *Dialectica*, vol.70, No. 4, pp. 531-547.
11. Russell, S., Norvig, P. (2010): *Artificial Intelligence: A Modern Approach*. Person, 3^{rd} edition.

Automatically Created Noun Explanations for English

Marie Stará

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

Abstract. In this paper, I comment on the automatically created explanations of word meaning for English nouns. These explanations are built using data gathered from Word Sketches created by a special Sketch Grammar.

Keywords: explanation, corpora, word sketch

1 Introduction

Following my work on automatic creation of dictionary definitions—or more precisely word meaning explanations—for Czech ([1], [2], [3]), I am modifying the method for English, so as to find out whether my approach is applicable in other languages. Hence, the purpose of this paper is to test the usability of the conversion on a smaller corpus, show the automatically created explanations of English nouns and evaluate the results.

The purpose of these explanations is to approximate the meaning of any given word by offering a set of hints (possibly) useful for understanding it.

2 Construction of Explanations

The explanations are created from the Word Sketches acquired using specially developed Sketch Grammar applied to the British National Corpus. Using Python script I take first three lemmata with the highest score for each relation and merge them in groups described below and demonstrated on the results for lemma house.

house
similar meaning can have (a/an) home, sale, garden, flat, shop, building, room
public, upper, big house
for example (a/an) home, hotel
can have/contain (a/an) garden, bedroom, room
(a/an) family, time, people can have/contain (a/an) house
is a subject of build, stand, belong

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 83–87, 2019.

is an object of terrace, buy, build
of (a/an) parliament, card, lord
with (a/an) garden, roof, wall

The first group of words contains hypernyms and (loose) synonyms, combinig results of five relations and data from Thesaurus (provided by Sketch Engine). These relations are: (1) *and_other* (two nouns connected by *and/or other/different/additional/further/more/such/next/similar*), (2) *WORD_is* (two nouns connected with lemma to be, possibly with other words (excluding nouns and verbs) in between), (3) *N_coord* (two nouns connected by and/or/neither-nor/either-or), and (4, 5) *hypo_hypero* (more complicated rules, basically two nouns connected either with *and (also) similar*, or *is type of*). These results are shown as a "similar meaning".

similar meaning can have (a/an) home, sale, garden, flat, shop, building, room

These similarities are followed by the list of most specific adjective modifiers of the given noun (relation *adj_modif*).

public, upper, big house

The next line of the explanation shows relation that occurs only in less than 2/3 of the test set; *for_example* (nouns connected by *(for/such) example/instance/e.g./as/like*).

for example (a/an) home, hotel

The following parts of the explanation are related to holonymy and meronymy (partitive). The fist part—"lemma can have/contain results"—is created combining relations *WORD_has* (basically two nouns connected by verb to have) and *consists of* (simply put, two nouns connected by *(can) consist/make/form/comprise/contain/include/incorporate/embody/involve/hold/cover (of)*). The second part—"results can have/contain lemma"—uses the relation *what_who_ has_WORD* (again basically two nouns connected by verb to have).

can have/contain (a/an) garden, bedroom, room
(a/an) family, time, people can have/contain (a/an) house

Other group of results is formed by verbs for which the given word is a *subject* or *object*, hence showing the results the results of relations *is_subject* and *is_object*, respectively.

is a subject of build, stand, belong
is an object of terrace, buy, build

Last two lines show results created primarily for Czech, which are also (a bit surprisingly) useful in English explanations: *of* (using the relation *gen*, two nouns connected by *of*) and *with* (sing the relation *instr*, two nouns connected by *with*).

of (a/an) parliament, card, lord
with (a/an) garden, roof, wall

3 Evaluation of Explanations

I evaluated the explanations on a set of 70 nouns; for the sake of comparison, I used the translation of my test set for Czech (hence I deleted words with multi-word translation as is, e.g. *fish breeding ground* (trdliště). Six of these words never occurred in the corpus; two words have frequency lower than twenty. Ten words have frequency in between 20 and 100. The highest frequency in the test set is 67 826 (*song*). It is apparent from these data that the test set contains words of various frequencies. Another distinction between the tested expressions is the presumed difficulty of creating the explanation. (It is rather straightforward to explain meaning of e.g. *a dog* (an animal which barks) or *a house* (a building for living). Explaining what a is e.g. *nothingness* (absence of anything?) or *laughter* (loud expression of happiness?) is supposedly more challenging, especially should the explanation be reasonably short and specific. This distinction is most visible in between abstract and concrete nouns.

The test set contains words with more meanings, homonyms (*(a) lead—to lead*, and synonyms (*couch*, *sofa*). Some words were picked ad hoc to ensure the test set is sufficiently differentiated.

3.1 Examples

As mentioned above, the test set contains lemmata with zero frequency in the corpus. Three words are interpreted as a different part of speech (*an expose, (a) lead* are recognised/appearing in the corpus only as verbs, *a mammoth* as an adjective).

There are also other words, for which the Word Sketches do not yield sufficient data, e.g. *mamluk* (frequency 35). Apart from this extreme case, there are other words without enough good data, e.g. *excavator*, where only the first two lines (and arguably the *is_object* relation) contain sensible result.

mamluk
is an object of exist

excavator
similar meaning can have (a/an) digger, tractor, shaker, extractor, servo, pride mini, rotary, bulk excavator

for example (a/an) bow, hall, town
can have/contain (a/an) coin, finance, right
(a/an) other can have/contain (a/an) excavator
is a subject of indemnify, assign, exploit
is an object of alert, prompt, protect
of (a/an) heart

There are not only quite good results (see *house* above) and rather bad ones, for the most part one explanation contain ballanced amount of good and bad (and okay-ish) results, e.g. *bed*.

bed
similar meaning can have (a/an) breakfast, border, table, room, door, wall
double, twin, unmade bed
for example (a/an) doctor, child
(a/an) price, kitchen, room can have/contain (a/an) bed
is a subject of stare, knit, sleep
is an object of share, wet, strip
of (a/an) rose, lettuce, nail
with (a/an) flu, someone, sheet

An often appearing problem is badly recognised part of speech in the data. This can be seen e.g. in the explanation of *oak*, where there are nouns recognised as verbs (e.g. *pine* in all occurrences in coordination *oak and pine*).

3.2 Evaluation

Generally, the best results are the adjective modifiers, followed by the word with similar meaning and the of-relation. Surprisingly, the explanations are more reliable when the given lemma is an object, not a subject of a verb.

Significantly fewer results are found for the has/contains relations, as well as the with-relation. These result might change when a bigger corpus is used. The least reliable results are the lists of examples, where there is a lot of noise.

As hinted in 3.1, one of the reasons the explanations contain irrelevant (or simply wrong) data is wrongly tagged tokens. Nevertheless, it is apparent the results show the chosen approach is applicable and can be used with minor editing.

4 Conclusions

With the corpus data containing mistakes in part of speech tags, it is quite difficult to automatically create sufficient explanation for any given word. The results are, nevertheless, encouraging as bigger data generally lead to better results.

Acknowledgements. This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin infrastructure LM2015071.

References

1. Stará, M. Automatická tvorba definic z korpusu. Masters thesis, Masaryk University (2019)
2. Stará, M. Automatically Created Noun Definitions for Czech. Proceedings of the Twelfth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2018 (2018)
3. Stará, M., Kovář, V. Options for Automatic Creation of Dictionary Definitions from Corpora. Tenth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2016 (2016)

The Concept of 'empire' in Russian and Czech

Victor Zakharov

Saint-Petersburg State University
Universitetskaya emb. 7-9
199034 Saint-Petersburg, Russia
v.zakharov@spbu.ru

Abstract. The paper is dealing with subject of forming semantic fields. The 'empire' semantic field in 2 languages (Russian, Czech) was chosen as an object of investigation. The paper describes a descriptive statistical method of forming semantic fields based on linguistic corpora. The result is a specific lexicographic product (distrubutive thesaurus) for each language with quantitative characteristics of the connectedness of lexical units. At the last step linguistic correlation between elements of these two thesauri is shown. The research is implemented on the basis of Sketch Engine and Czech National Corpus. In the aspect of theory, we get a fragment of the semantic description of Russian and Czech languages and a description of new methods for analyzing vocabulary and semantics of the language.

Keywords: concept, semantic field, concept of empire, distributional thesaurus, Russian, Czech, corpora

1 Introduction

The subject of the study is to consider the concept of 'empire' in Russian and Czech. We mean here the term, by which one determines the content plan of the word, i.e. the notion, fixed in the language and correlated with other notions associated with it. Our task is to reveal the lexical content of these interconnected notions defined by the named concept.

Concepts underlie what linguists and cognitologists call the linguistic image of the world. This is a set of ideas about the world historically formed in the everyday consciousness of a given community and reflected in the language, in other words, this is conceptualization of reality. The linguistic image of the world determines the various aspects of the language, its vocabulary, its capability to generate words, to influence the syntax of phrases and sentences, as well as paremiological layer of language. Only linguistic images of the world in specific national languages really exist and can be analyzed, this is national linguistic images of the world. The linguistic image of the world is time-varying. In this work, we are interested in the current state of the language.

The set of lexical units each of which has some common component of the meaning forms a semantic field. The field is characterized by the presence of an inventory of elements connected by systemic relationships. It has a central

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 89–97, 2019.

part, the core, the elements of which have a complete set of features that define this grouping, and the periphery, elements of which do not have all the features characteristic of the field. The field implies the continuity of the connections of set units. Fields are characterized by the possibility of quantitative expression of the strength of relations between field members.

The choice of Russian and Czech languages is due to the fact that in both languages the concept of 'empire' is strongly connected with the historical memory of the people and that it is "alive" in the linguistic consciousness of native speakers.

2 Statement of the problem and research methodology

Unlike of psycholinguistics the task of computational linguistics is automatic selection of lexical units for semantic fields. The method uses distributional statistical analysis based on linguistic corpora. The corpus approach, however, does not exclude the subsequent involvement of expert knowledge.

The objective of our study is to create two associative thesauri with quantitative characteristics of lexical units for two languages with examples from corpora. In this paper, we solve the problem of selecting the vocabulary of the empire semantic field in each language, getting statistical characteristics of lexical units on the basis of corpora, and the identification of Russian-Czech (Czech-Russian) translation equivalents of semantic field units.

There are two aspects of functioning of a linguistic unit, syntagmatics and paradigmatics. The methodology of the study is a corpus-oriented analysis of the paradigmatics and syntagmatics of lexical units, which form the semantic field for the word empire. Our materials are corpora with linguistic tagging and corpus linguistic processors. At the same time, the other lexicographic resources might be included in the analysis if necessary.

Corpus linguistics made it possible to "calculate" different types of compatibility which are combined under the term multiword expressions. But if syntagmatic relations are explicitly presented in text and can be extracted from it on the basis of a linear sequence, paradigms are hidden and it requires extralinguistic knowledge and/or sophisticated procedures should be developed to extract them from texts.

The building of semantic field is the task of modeling the conceptual subsystem of a language. Since our knowledge of the world is reflected in texts, we can set the task of extracting a system of concepts from texts. In this paper, we try to extract interrelated units grouping around the core notion of empire, starting from keywords that most closely express the meaning of the core concept.

Already at the dawn of computational linguistics, the idea was put forward that paradigmatic connections could be deduced from syntagmatic connections. The principle of the transition from the study of textual (syntagmatic) links to systemic (paradigmatic) underlies various distribution and statistical techniques

[6, 9, 10]. It was believed that two elements were connected paradigmatically if both of them are textually systematically connected with some third elements.

However, the capabilities of computational technology for a long time did not allow to put these ideas into practice. In order to talk about the regularity of any statistical distributions, very large data sets are needed. That became possible only with the development of the web and the creation of large text corpora. At the same time, appropriate software tools appeared [1, 4, 8]. Attention was drawn to the fact that it was also important to take into account the occurrence of a syntactic relationship between contextually close elements of the text [2, 5].

3 Research material and tools

In this work, the Sketch Engine system (https://app.sketchengine.eu) was mainly used for the research. We used ruTenTen 2011 corpus and csTenTen 2017 corpus and also we used the Czech National corpus (ČNK) (syn v7 and Treq).

The advantage of the Sketch Engine for our purposes is its special tools that make possible distributional statistical analysis, they are "Thesaurus" (building a distributional thesaurus) and "Clustering" (grouping of thesaurus units in clusters, i.e. lexical-semantic groups).

The thesaurus in Sketch Engine allows to see which words have a similar distribution with the given word, which, as a rule, is caused by their semantic proximity, i.e., in fact, this tool forms a uniterm semantic field. Word distribution similarity is calculated statistically, calculation is based on the association measure logDice [7] and lexical-syntactic patterns [3]. In the next step the inclusion in the semantic field the characteristic stable phrases is provided by the Collocations tool.

4 Technology of formation of the core of the empire semantic field

At the first stage, various lexicographic sources were used to describe the concept of empire in terms of keywords. Analysis of dictionary definitions from various Russian and Czech dictionaries made it possible to identify the main meanings and, respectively, semantic attributes of the concept of empire:

1) monarchy, headed by the emperor;

2) large state, consisting of several parts, possibly colonies;

3) metaphoric meanings derived from one of the first two (e.g. a large enterprise, parts of the natural world, etc.).

In our analysis, we deal only with vocabulary related to the first concept.

A technology of formation of semantic fields based on the diachronic approach was developed and tested, then data from text corpora printed in different historical periods were analyzed. For a detailed account, see [12]. In this paper, we are interested in a synchronous approach, how the concept of empire in modern Russian and Czech texts is implemented.

As a result of a definitional analysis of explanatory dictionaries and dictionaries of synonymy, elementary units of a meaningful plan were identified, 10 lexemes in each language. In doing so, we sought that these terms be monosemic.

Lexical identifiers of the concept of empire in Russian are as follows: государь (sovereign), держава (power), династия (dynasty), император (emperor), императрица (empress), империя (empire), монарх (monarch), монархия (monarchy), правитель (ruler), самодержавие (autocracy). Lexical identifiers of the concept of empire in Czech are as follows: *císař* (the emperor), *císařství* (empire). *dynastie* (dynasty), *impérium* (empire), *král* (king), *mocnářství* (monarchy), *monarchie* (monarchy), *panovník* (ruler), *říše* (empire), *vládce* (ruler).

Then for each of them 10 distributional thesauri were built in Sketch Engine on the basis of ruTenTen 2011 and csTenTen 2017 corpora (Fig. 1). In order to avoid getting into the resulting field of nonrelevant vocabulary the volume of the distributional thesaurus was limited to 15.

císař (noun)
Czech Web 2017

Lemma	Score	Freq
král	0.373	975,654
panovník	0.350	88,279
papež	0.348	208,554
kníže	0.321	151,336
vůdce	0.296	300,593
královna	0.294	249,985
vládce	0.292	119,743
biskup	0.279	231,650
prezident	0.276	1,510,494
generál	0.265	216,111
bratr	0.262	773,133
velitel	0.259	310,254
otec	0.254	1,384,487
ministr	0.250	1,316,050
premiér	0.248	493,872

Fig. 1: The distributional thesaurus (semantic field) for the word *říše*

The important characteristics here are the coefficient of the semantic proximity of the lexemes with a headword (score) and their frequency (freq).

We can suggest the language homogeneity in the selected corpora. Both of them are created on the base of texts from web, and contain mainly modern texts, both corpora are created using the same technology. We can say that they contain the vocabulary of the modern language and thereby reflect the modern state of linguistic consciousness.

On next stage, all 10 thesauri were put together into one dataset. Moreover, for each term, the average score was calculated. The assumption was made empirically that if a lexeme occurs in at least N thesauri (we call N the stability

coefficient), it is a candidate for inclusion in the core of the semantic field. The lexemes with value of the score less than N form its periphery. Both in the center and in the periphery area the lexemes can be sorted according to their score.

Further, for each element of the field core the most characteristic bigram collocations were identified using ČNK syn v7 corpus and Collocations tool. Bigrams were sorted by the MI.log_f association measure as one of the most effective ones.

5 The results obtained

5.1 Empire semantic field in modern Russian

The intersection of 10 thesauri (150 lexical units) yielded 79 unique lexemes, of which 17 met 3 or more times (in 3 or more thesauri), 19 - 2 times and 43 once. 17 units that have occurred 3 or more times form the core of the empire semantic field. They are as follows (in alphabetic order): владыка (lord), вождь (leader), государственность (statehood), государь (sovereign), держава (power), династия (dynasty), император (emperor), императрица (empress), империя (empire), князь (prince), король (king), монарх (monarch), монархия (monarchy), папа (pope), правитель (ruler), принц (prince), самодержавие (autocracy), царство (kingdom), царь (tsar), цивилизация (civilization).

It is interesting to note that the initial lexical identifiers of the concept of empire, which we took from dictionaries, appeared in the consolidated distributional thesaurus only 2 times (*power* and *dynasty*) and 1 time (*autocracy*). We have included them in the core for now. But since we consider our corpora as a model of a modern language, we can say with caution that these concepts are gradually leaving the concept of empire.

Perhaps the explanation of the appearance in this list the polysemous word папа (in conversational Russian 'dad') requires clarification. An analysis of corpus contexts showed that it was about the concept of the Pope of Rome which has a close connotation with the monarchs.

The periphery of the field includes 59 lexemes such as абсолютизм (absolutism), Англия (England), аристократия (aristocracy), властитель (lord, sovereign), Германия (Germany), государство (state) etc.

Also collocations will be added to the empire semantic fields both in Russian and in Czech.

5.2 Empire semantic field in modern Czech

The intersection of 10 thesauri (in total 150 lexical units) gave 88 unique lexemes, of which 13 met 3 or more times, 20 - 2 times and 55 once. If we take the stability coefficient equal to 3, then 13 lexemes form the core of the empire semantic field for the Czech language. Interestingly, for the Czech language, three of the original identifiers of the concept of empire which we took from Czech dictionaries were found in the combined distributional thesaurus for the Czech

language only 2 times (*císařství*, *dynastie*, *mocnářství*). However, we included them in the core of the semantic field for the Czech language.

The full list of the core of the empire semantic field for the Czech language is as follows (in alphabetic order): *císař* (emperor), *císařství* (empire), *dynastie* (dynasty), generál (general), impérium, impérium (empire), kníže (prince), král (king), *královna* (queen), *království* (kingdom), *mocnářství* (monarchy), *monarchie* (monarchy), *panovník* (ruler), *říše* (empire), *velitel* (commander), *vládce* (ruler), *vůdce* (leader). The periphery of the field includes 72 lexemes.

5.3 Comparison of the core of the empire semantic field in Russian and Czech

Let's try to compare the filling of the empire sematic field in Russian and Czech. If we temporarily exclude from consideration lexemes that mean roughly the same in Russian and Czech and lexemes that are present only in one of the language fields (государственность (statehood), папа (pope), князь (prince), цивилизация (civilization), generál (general), *velitel* (commander)), then lexemes related to the two microfields will remain.

The first microfield contains different names for the concept of empire: in Russian they are империя (empire), царство (kingdom), держава (power), partly монархия (monarchy); in Czech *impérium* (empire), *říše* (empire), *království* (kingdom), *císařství* (empire), *mocnářství* (monarchy), partly *monarchie* (monarchy). The second microfield contains different names for the concept of emperor: in Russian, they are монарх (monarch), правитель (ruler), царь (tsar), владыка (ruler), государь (sovereign), император (emperor), императрица (empress); in Czech *panovník* (ruler), *vládce* (ruler), *císař* (emperor), *král* (king), *královna* (queen). A few more words can be added to these microfields from peripheral vocabulary.

If we approach the analysis of these microfields from the point of view of historical science, we can show the national-cultural and historical conventionality and feature of each term in each language. However, we are interested in their relationship in two languages from the point of view of ordinary language consciousness. We can say that two ways to put together semantically similar terms in different languages are bilingual dictionaries and examples of translation.

6 Translation equivalents of lexemes of the empire sematic field in Russian and Czech

The last stage of the work is study of interlanguage equivalents. A preliminary assessment was carried out on the base of 2-volume dictionaries edited by L.V. Kopecky (Russian-Czech, Czech-Russian). Vocabulary equivalents can be seen in the left column in Table 1. When analyzing translation dictionaries, we cannot say with what probability one or another equivalent is used.

It is interesting to see which words (and why?) will prevail when translating the same concept. For example, the Czech "říše" in Russian can sound like империя (empire), королевство (kingdom), царство (kingdom), рейх (Reich),

Германия (Germany). The Russian империя (empire) can be translated into Czech as *impérium*, *říše*, *císařství*, *država* and others. The same applies to other terms, too.

Using the terms from our semantic field as an example, we made an attempt to evaluate this using the InterCorp parallel corpus that is a part of ČNK. ČNK programmers developed the Treq tool on the basis of the InterCorp [11], which allows to get all the translations of a given word and statistics on the frequency of translation equivalents that were found in the corpus.

The results obtained (for the lack of place only for translations from Czech to Russian) are shown in Table 1. The left column contains a word in the input language with a translation from the dictionary, the top row contains words of the output language (translations). In cells, quantitative characteristics of translated equivalents are given: the upper number is the number of translations for a given pair of words encountered in the InterCorp corpus, the lower number is the percentage of this translation from all translations of this word (the percentage value is rounded). Rare and erroneous cases are not included, so percent sum is not always 100%. The most frequent translations are highlighted in bold.

Table 1: Translation equivalents for words from the core of the empire semantic field for the Czech language according the InterCorp corpus

	империя	царство	держава	рейх	королевство	монархия	владение	метрополия	государство
říše империя царство	200 51%	56 14%	4 1%	50 13%	37 10%		4 1%		10 2.5%
impérium империя	230 97%								
království королевство		61 20%			216 70%	1 0.3%			
císařství империя	6 86%							1 14%	
mocnářství монархия			1 8%			12 92%			
država владение	1 6%		2 12%				7 44%		1 6%
carství царство		3 75%							1 25%
monarchie монархия						68 97%			

In dictionaries, usually only the main translation is given, and it is usually the most frequent in corpus, but the number of translation equivalents in real

texts is greater (see, for example, the translations for říše) and we see their ratio, too.

7 Conclusion

We see that the use of text corpora and "smart" corpus instruments allows one to identify syntagmatic and paradigmatic connections in an automated mode and create an adequate filling of the term system, in this case it is the semantic field that describes the concept of empire. Lists of words were obtained, greatly expanding available lexicographic manuals.

Finally, it can be stated that the task of building one small semantic field reflects the peculiarities of the lexico-semantic system of a language as well as opportunities and barriers in automation of semantic processing.

Acknowledgments. This work was implemented with financial support of the Russian Foundation for Basic Research, Project No. 18-012-00474 "Semantic field 'empire' in Russian, English and Czech" and partly by Project No. 17-04-00552 "Parametric modeling of the lexical system of the modern Russian literary language".

References

1. Blancafort, H. Daille, B., Gornostay, T., Heid, U., Méchoulam, C., Sharoff, S.: TTC: Terminology extraction, translation tools and comparable corpora. In: 14th EURALEX International Congress, pp. 263-268 (2010)
2. Gamallo, P., Gasperin, C., Augustini, A., Lopes, G. P.: Syntactic-Based Methods for Measuring Word Similarity. In: Text, Speech and Dialogue: Fourth International Conference TSD–2001. LNAI 2166, Springer-Verlag, pp. 116–125 (2001)
3. Kilgarriff, A., Rychly, P.: An efficient algorithm for building a distributional thesaurus (and other Sketch Engine developments). In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions. Czech Republic, June 2007, pp. 41–44 (2007)
4. Kilgarriff, A., Rychly, P., Smrz, P., Tugwell, D.: The Sketch Engine. In: Proceedings of the XIth Euralex International Congress, Lorient: Universite de Bretagne-Sud, pp. 105–116 (2004)
5. Pazienza, M., Pennacchiotti, M., Zanzotto, F.: Terminology extraction: an analysis of linguistic and statistical approaches. In: Knowledge Mining Series: Studies in Fuzziness and Soft Computing, Springer Verlag, Berlin, pp. 255–279 (2005)
6. Pekar, V.: Linguistic Preprocessing for Distributional Classification of Words. In: Proceedings of the COLING–04 Workshop on Enhancing and Using Electronic Dictionaries, Geneva, pp. 15–21 (2004)
7. Rychlý, P.: A lexicographer-friendly association score. In: Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN, Brno, pp. 6–9 (2008)
8. Sharoff, S.: Open-source corpora: Using the net to fish for linguistic data. In: International journal of corpus linguistics, John Benjamins Publishing Company, Vol. 11, No. 4, pp. 435–462 (2006)

9. Shaykevich, A.Ya.: The distributive and statistical analysis in semantics [Distributivno-statisticheskij analiz v semantike]. In: Principles and methods of semantic researches [Principy i metody semanticheskih issledovanij], Moscow, pp. 353-378 (1976)
10. Smrž, P., Rychlý, P.: Finding Semantically Related Words in Large Corpora. In: Text, Speech and Dialogue: Fourth International Conference (TSD–2001), LNAI 2166, Springer-Verlag, pp. 108–115 (2001)
11. Škrabal, M., Vavřín, M. The Translation Equivalents Database (Treq) as a Lexicographer's Aid. In: Lexical Computing CZ s. r. o. Electronic lexicography in the 21st century. Proceedings of eLex 2017 conference (2017)
12. Zakharov, V. Ways of automatic identification of words belonging to semantic field. In: Jazykovedný časopis. 2019. Vol. 70. No. 2, pp. 234–243 (2019)

Czech Question Answering with Extended SQAD v3.0 Benchmark Dataset

Radoslav Sabol, Marek Medveď, and Aleš Horák

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00, Brno, Czech republic
{xsabol, xmedved1, hales}@fi.muni.cz

Abstract. In this paper, we introduce a new version of the Simple Question Answering Databases (SQAD). The main asset of the new version lies in increasing the number of records to a total of 13,473 records. Besides the database enlargement, the new version incorporates new restrictions of specifying different formats of the expected answer for a given question. These new restrictions are connected with automatic database consistency checks where new sub-processes safeguard the database correctness and consistency.
We also introduce a new on-line annotation tool used which offered a unified environment for extending the SQAD data in a crowdsourcing experiment.

Keywords: question answering, QA benchmark dataset, SQAD, Czech

1 Introduction

The evaluation of question answering (QA) results has substantially improved in recent years. Detailed comparison of new state-of-the-art QA tools is now possible thanks to large shared benchmark datasets, especially for English. These datasets consists of thousands of records, which makes them an important resource in the QA field for both evaluation and training of unsupervised approaches.

The Stanford Question Answering Dataset (SQuAD [6]) is one of the best-known QA benchmark dataset. SQuAD consists of more than 100,000 questions with several correct answers for each question. The state-of-the-art tools reach more than 92% F1 score with this dataset.

The ReAding Comprehension from Examinations (RACE [3]) dataset is another benchmark dataset for the QA task which also consists of nearly 100,000 questions where each question has 4 candidate answers. Current evaluation result using this dataset reach nearly 90% F1 score.

Majority of current state-of-the-art QA tools use unsupervised machine learning approaches which make them dependent on large dataset such are those mentioned above. Without such large datasets, the unsupervised models are not able to provide good generalizations of the problem. A problem arises

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 99–108, 2019.

when the successful approaches are to be applied to a non-mainstream language for which a corresponding dataset is unavailable.

In this paper, a new version of the Czech benchmark QA dataset called SQAD (Simple Question Answering Database) is introduced. The latest version is numbered 3.0 and consists of almost 13,500 question-answer pairs. Besides the enlargement of the dataset, the format of the record items is enhanced to satisfy two main requirements. The answer selection item[1] must consist of a single sentence from the text and this sentence must contain the expected answer. The answer extraction item (the expected exact answer) has to be a proper sub-phrase of the answer selection sentence. This new restrictions ensure database consistency and allow for automatic database correctness and consistency checks.

During the preparation of the new SQAD version, a new online annotation tool, named AddQA, for unifying the process of creating new SQAD records was developed.[2] This tool implements multiple sub-processes. Firstly, the indicated article content is downloaded and preprocessed from the Czech Wikipedia dump. Secondly, the selected text parts (questions, answer selection and exact answer) and the whole text are automatically annotated with morphological information and part-of-speech tags.

In the evaluation section, the final statistics of the new SQAD v3.0 database and the current results of the AQA [4] system with this new SQAD version are presented.

2 SQAD Version 3.0

The new version 3.0 of the Czech QA benchmark dataset SQAD features a substantial enlargement of SQAD v2.1 introduced in [8]. All SQAD versions comprise both the carefully processed manual annotations of questions and answers by human annotators and the database accompanying tools which automatically preprocess, store and check the data throughout the whole development process. From version to version, the tools are adapted to improve the uniqueness and consistency of the annotated data. In version 3.0, new web interface for annotators called AddQA was implemented and several new consistency checks were added to make the data preparation process more streamlined and less time consuming.

In the first part of this section, we describe new web based interface for annotators and in the second part we introduce rules (restrictions) according to which the new records have to be created.

2.1 New AddQA Online Annotation Tool

The process of adding new records to the SQAD dataset is quite complicated and multilevel. For a substantial increase in the number of the dataset records

[1] The sentence which contains the expected answer to the given question.

[2] All new records in SQAD v3.0 have been created by this new tool.

Remaining annotations: 198 of 200

Question type	Missing QA	
ABBREVIATION	**10** of 10	Add new ABBREVIATION QA
ADJ_PHRASE	**50** of 50	Add new ADJ_PHRASE QA
CLAUSE	**10** of 10	Add new CLAUSE QA
DATETIME	**9** of 10	Add new DATETIME QA
ENTITY	**28** of 28	Add new ENTITY QA
LOCATION	**9** of 10	Add new LOCATION QA
NUMERIC	**4** of 4	Add new NUMERIC QA
PERSON	**28** of 28	Add new PERSON QA
VERB_PHRASE	**50** of 50	Add new VERB_PHRASE QA

Already annotated: 2 of 200

Question ID	Q type	A type	Question	Answer	Sentence	URL
19	LOCATION	LOCATION	Kde se nachází Kuba?	v severním Karibiku	Kuba se nachází v severním Karibiku a její břehy omývají Karibské moře, Mexický záliv a Atlantský oceán.	https://cs.wikipedia.org/wiki/Kuba
21	DATETIME	DATETIME	Kdy se narodil Jeremy Clarkson?	11. dubna 1960	Jeremy Clarkson (* 11. dubna 1960), celým jménem Jeremy Charles Robert Clarkson, je anglický hlasatel, žurnalista a spisovatel, který se specializuje na motorismus.	https://cs.wikipedia.org/wiki/Jeremy_Clarkson

Fig. 1: AddQA overview page: display of the question types to create and a listing of previously created records.

(question-answer pairs), a new online interface, denoted as AddQA, was developed where the annotators do not deal with routine work (split text to sentences, annotate words, etc.) and concentrate only on the expert annotations that can not be automated. The AddQA process consists of multiple steps that lead to a final new SQAD record. As it was introduced in [8,2], each SQAD record consists of several files:

- the *question*
- the *full article* text
- the *answer selection* sentence which contains the expected answer as a sub-phrase
- the *answer extraction* result, i.e. the exact expected answer
- the *URL* of the original article in the Czech Wikipedia
- the QA *metadata* – the question and answer types

When entering all this information, each annotator has to go through the following steps:

- Create a new record or to edit a previously created one. In order to create a well balanced dataset, each annotator was assigned a predefined composition of the requested question type classes to add. The annotator thus knows in advance what kind of questions is needed and he or she can pick up a suitable article from Wikipedia. This is implemented in the first annotation phase displayed in Figure 1. The top part of the image informs the user how much records of each question type has to be added and the bottom part lists all records created by the user until now. Besides creating a new record, the possibility to adjust previously created question or answer is accessible from this page.

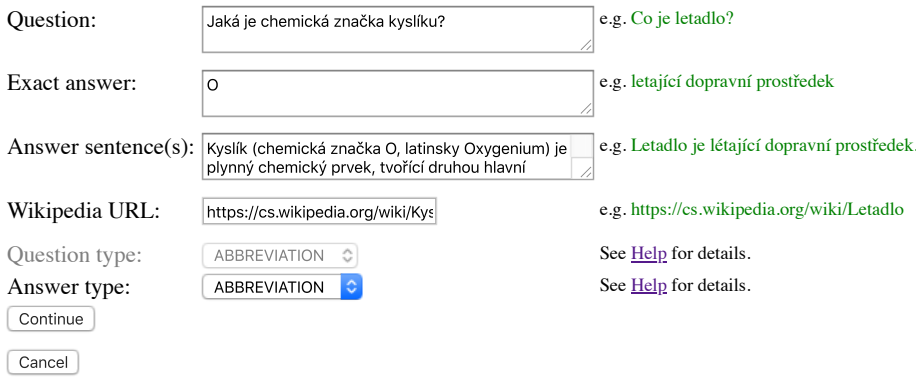


Fig. 2: The new record form for the question "Jaká je chemická značka kyslíku (What is the chemical symbol of oxygen)?"

- Fill the (new) record form. After selecting a predefined question type to add, the annotator is navigated to editing form. Here, the corresponding question information is connected to a text (the full document, the answer sentence and the exact answer) from a chosen Wikipedia article identified by its URL. The user also enters the type of the expected answer. An example for an ABBREVIATION question is presented in Figure 2.
- Final check of record correctness. When the filled form is submitted, two main sub-processes are triggered. The first one automatically uses the Wikipedia API to download the raw representation of the article. The second process takes care about an automatic annotation of sentence boundaries, lemmata, morphological categories and part-of-speech tags by the Unitok [5], Majka [7] and Desamb [9] tools. The final record is displayed in Figure 3.

The last but very important part of web annotation tool is help page where annotators can find description of question and answer types and see demo examples for each type to get main idea how to form a new one (see Figure 4).

3 SQAD Format Update

To be able to automatically check the dataset consistency, a set of rules for each new record has been defined. In the current version, these rules have been enforced by a semi-automatic batch processing. In a new AddQA version, they are planned to be incorporated in a form of an automatic sub-process.

The main goal of these restrictions is to create a coherent and consistent database. The first two restrictions are focusing on the answer selection part, the third rule handles the format of the expected answer.

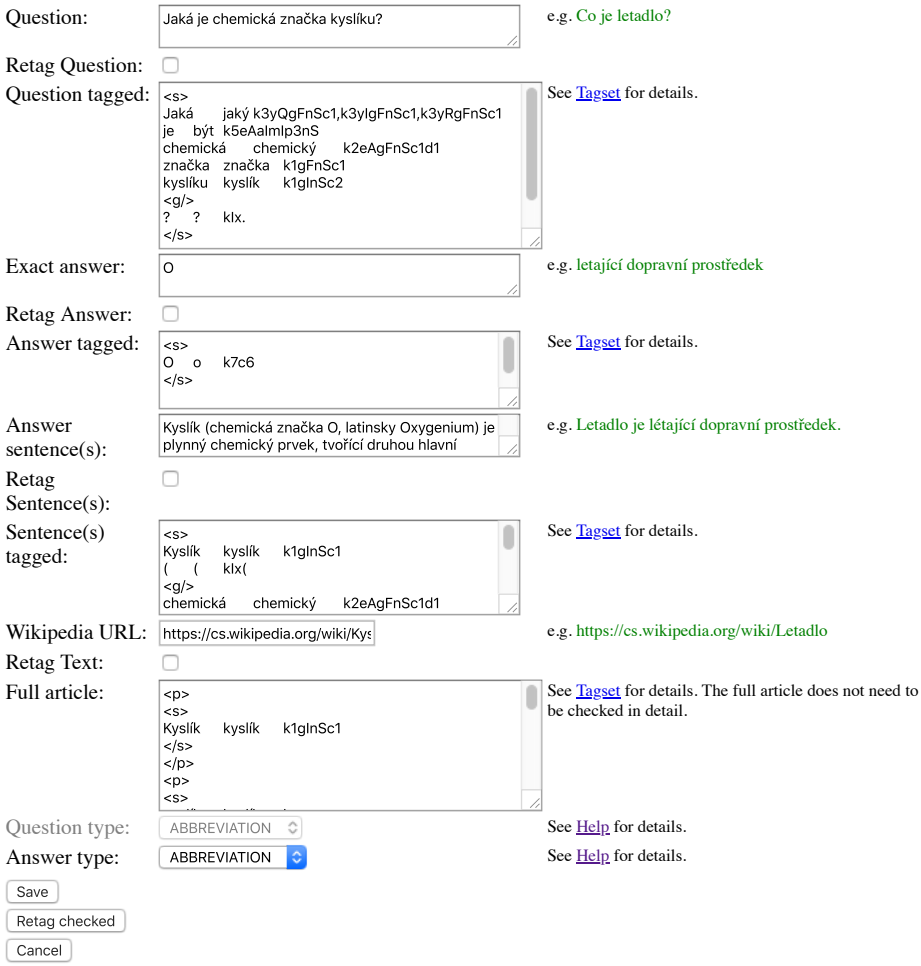


Fig. 3: Final check of record correctness.

3.1 The Answer Selection Format

A Wikipedia article is a conventional text therefore there are naturally connecting anaphoric references such as "*Peter was a famous singer. He was also a famous English song writer.*" Here "*Peter*" is an antecedent[3] of the pronoun "*He*". In case an annotator creates a question "*What is the name of a famous English song writer?*", an issue of choosing the of correct answer selection sentence arises. In previous versions of the SQAD database, such case could be annotated in three possible ways. The first option was to mark just the sentence containing the exact

[3] the target of an anaphoric reference

Question type	Description	Example question	Example Answer
ABBREVIATION	The question asks for abbreviation of some name.	Jakou chemickou značku má vápník?	Ca
ADJ_PHRASE	Question asks about some specific group of things, that is usually specified by adjective.	Jaká je tradiční barva Oxfordské univerzity?	tmavě modrá
CLAUSE	Question is general and the answer can be any general clause.	Proč se chtěla Marie Terezie spojit s Francií?	Protože by Prusko ztratilo svého důležitého spojence
DATETIME	Main goal of question is to determine certain point in time.	Kdy se narodila Petr Kvitová?	8. března 1990
ENTITY	Main goal of question is to name a thing (not a person) that meets all conditions of question.	Jak se jmenuje největší planeta sluneční soustavy?	Jupiter
LOCATION	Main goal of question is to determine certain place.	Kde zemřel Josef Kajetán Tyl?	Plzeň
NUMERIC	Main goal of question is to determine certain number.	Do kolika větví je rozdělena Armáda České republiky?	Do tří
PERSON	Main goal of question is to name a person that meets all conditions of question.	Kdo byl 33. prezidentem Spojených států amerických?	Harry S. Truman
VERB_PHRASE	Main goal of question is to find out if something happened. The answer can be general verb phrase or confirmation in form of YES/NO.	Je Brno sídlem fotbalového týmu Bohemians 1905?	ne

Answer type	Description	Example question	Example Answer
ABBREVIATION	The answer is abbreviation of some name.	Jakou chemickou značku má vápník?	Ca
DATETIME	Answer is a certain point in time.	Kdy se narodila Petr Kvitová?	8. března 1990
ENTITY	Answer is a name of some thing (not a person).	Jak se jmenuje největší planeta sluneční soustavy?	Jupiter
LOCATION	Answer denotes a certain place.	Kde zemřel Josef Kajetán Tyl?	Plzeň
NUMERIC	Answer is a number.	Do kolika větví je rozdělena Armáda České republiky?	Do tří
ORGANIZATION	Answer is name of some organisation, band, company ...	Frontman jaké kapely je Jarda Svoboda?	Traband
OTHER	Answer is a general phrase that is not belong to other answer type	Co je hard rock?	hudební styl
PERSON	Answer is a name a person that meets all conditions of question.	Kdo byl 33. prezidentem Spojených států amerických?	Harry S. Truman
DENOTATION	Answer is a general name of a field, area or an approach.	Jak se nazývá obor, který se zabývá studiem chování živočichů?	Etologie
YES/NO	The answer is YES or NO.	Je Brno sídlem fotbalového týmu Bohemians 1905?	ne

Fig. 4: Question and answer type description for annotators

answer[4] (in this case "*Peter was a famous singer*"). The second possible answer selection contained the sentence which corresponds directly to the question but the answer is just referred by the anaphora (for the example "*He was also a famous English song writer*"). And the third possibility annotated both these sentences as the answer selection. That is why this process has been unified in SQAD 3.0 with the following three rules:

1. The answer selection must contain exactly one sentence.
2. The answer selection sentence must contain the expected answer (as a sub-phrase).
3. If multiple sentences (with anaphora) are needed to uniquely identify the answer to the question, the answer sentence and the respective antecedent sentence(s) are stored in a new "question_context" file.

3.2 The Expected Answer Format

After choosing the answer selection sentence, the annotator is to demarcate the expected answer (answer extraction in the SQAD terminology) as a part of the annotated sentence. The answer should be as short as possible but still contain enough information to answer the question. In previous SQAD versions, the

[4] The shortest answer for the given question. In this example, the exact answer is "*Peter*".

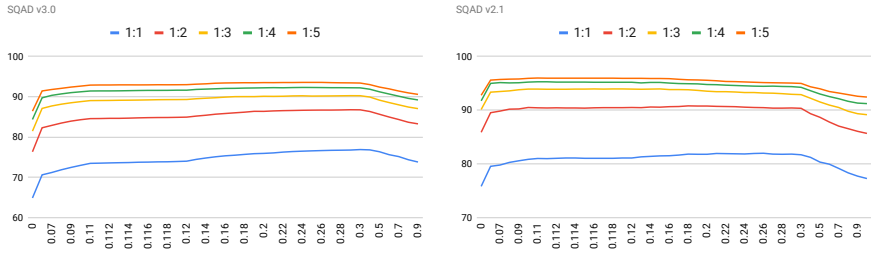


Fig. 5: Comparison of SQAD v2.1 and SQAD v3.0 on the document selection level (combined score on top 5 documents)

expected answer sometimes just "followed" from the answer sentence, but it was not a concrete sub-phrase of it. For the same reasons of consistency as mentioned in the previous section, the exact answer is now expressed in two forms. The original (expected) *answer* continues to be in the form as being formulated by a human after reading the given question and the answer selection (plus possible context) sentence(s). A new "answer_extraction" file now contains the real exact sub-phrase of the answer selection sentence (in the same form as stated in the text) and the *answer* should be its reformulation (or just a copy). This allows for extra automatic consistency checks of the answer extraction annotation.

Table 1: SQAD v3 statistics

	SQAD v3.0	SQAD v2.1
No. of records	13,473	8,566
No. of different articles	6,571	3,930
No. of tokens (words)	28,825,824	20,288,297
No. of answer contexts	378	0

Q-Type statistics:	SQAD v3.0	SQAD v2.1
DATETIME	14.7 %	21.6 %
PERSON	13.1 %	11.9 %
VERB_PHRASE	16.8 %	10.97 %
ADJ_PHRASE	11.2 %	2.7 %
ENTITY	18.4 %	20.4 %
CLAUSE	3.5 %	2.8 %
NUMERIC	7.3 %	10.7 %
LOCATION	12.4 %	17.8 %
ABBREVIATION	2.5 %	0.95 %
OTHER	0.1 %	0.18 %

A-Type statistics:	SQAD v3.0	SQAD v2.1
DATETIME	14.6 %	21.5 %
PERSON	13.2 %	12.3 %
YES_NO	16.8 %	10.95 %
OTHER	16.7 %	9.6 %
ENTITY	13.1 %	12.7 %
NUMERIC	7.4 %	10.7 %
LOCATION	12.3 %	17.6 %
ABBREVIATION	2.4 %	0.96 %
ORGANIZATION	2.1 %	2.5 %
DENOTATION	1.4 %	1.2 %

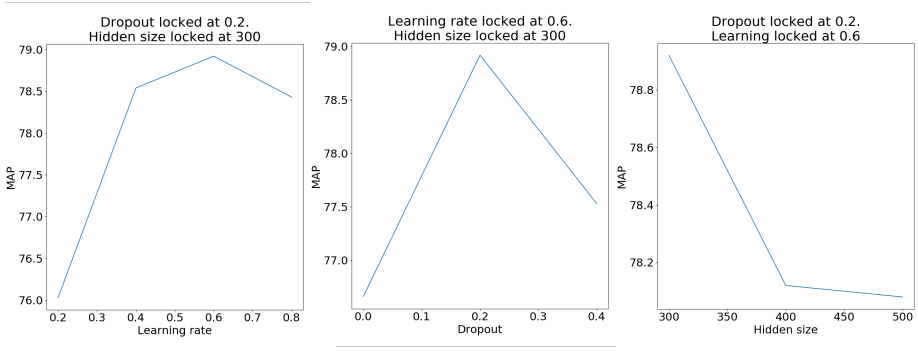


Fig. 6: The sensitivity graphs of the hidden size, dropout and learning rate hyperparameters in the SQADv3 evaluation.

4 Evaluation

The new SQAD version 3.0 is larger than all previous versions. It contains 13,473 records (question-answer pairs) which is almost 5,000 more than SQAD v2.1. For fine-grained statistics about the new version see Table 1.

The first tests of the Automatic Question Answering (AQA [4]) tool evaluated separately the document selection and the answer selection modules. The results of the document selection module are graphically illustrated and compared to the previous version in Figure 5.

The parameters of the AQA answer selection were adjusted according to their performance with SQADv2. The experiments included hidden representation vector dimensions of 300, 400, and 500 (400 was the most successful with the previous version of the dataset). The dropout values ranged for 0 (no dropout), 0.2 and 0.4 (0.6 was omitted for previous low performance). The learning rate range was extended to cover the values of 0.6 and 0.8, due to fact that 0.4 was the

Table 2: The SQADv3 answer selection accuracy per question and answer types

Question type	Count	MAP (%)	Answer type	Count	MAP (%)
VERB_PHRASE	546	80.06	YES_NO	539	79.73
NUMERIC	212	74.13	NUMERIC	215	73.63
ADJ_PHRASE	363	79.08	OTHER	526	76.79
CLASUE	99	67.81	DATETIME	470	81.88
DATETIME	473	82.12	LOCATION	415	84.87
ABBREVIATION	71	78.89	ENTITY	403	76.47
LOCATION	417	84.76	PERSON	421	77.53
ENTITY	571	77.16	ABBREVIATION	66	78.57
PERSON	418	77.41	ORGANIZATION	65	75.58
OTHER	1	33.33	DENOTATION	51	87.93

Table 3: Answer selection Precision at k ($P@k$)

Position k	Count	$P@k$	Position k	Count	$P@k$
1.	3,171	79.00%	6.	31	0.77%
2.	377	9.39%	7.	20	0.50%
3.	125	3.11%	8.	16	0.40%
4.	66	1.64%	9.	13	0.32%
5.	59	1.47%	$\geq$ 10.	136	3.40%

best and also the maximum value for previous runs, so the current experiments combined the learning rates of 0.2, 0.4, 0.6, and 0.8.

The training and evaluation procedure remained the same as in SQADv2. For this purpose, SQADv3 was partitioned into the train, validation, and test sets with the ratios of 60:10:30. All these sets comprise balanced proportions of the question and answer types. The answer selection models were trained for 25 epochs using Stochastic Gradient Descend [1] process, where the weights with the best performance with the validation set were applied to the test set. All 36 possible combinations of parameters were evaluated three times, which means that overall 108 models were trained. The best parameter combination for SQADv3 had the hidden size value of 300, the dropout value of 0.2, and the initial learning rate of 0.6. The Mean Average Precision (MAP) of this combination with the test set was **78.92%**, and the Mean Reciprocal Rank (MRR) of **85.95**. The sensitivity of various model parameters can be found in Figure 6. Table 2 shows the accuracy results per question/answer types. Precision at position k ($P@k$) of one of the three best performing models can be seen in Table 3.

To evaluate the dataset independently of the predefined train-validate-test split, 5-fold cross validation test were run with both SQADv2 and SQADv3 with the results presented in Table 4. The technique involves splitting the dataset to n (in this case n=5) equally sized partitions. For each of n runs, one of the partitions is labeled as test set with its own validation set (300 questions) while all other serve as the training set. The overall MAP is then computed as the average value of all n trained models reaching **77.68** which is a 0.51% improvement when compared to SQADv2. Th 5-fold cross validation was performed only with the best parameter combination for both versions of the dataset (using Adadelta optimizer).

Table 4: Results of 5-fold cross validation for both versions of the SQAD dataset

Test partition no.	SQADv3.0 MAP	SQADv2.1 MAP
1	82.76%	87.16%
2	74.26%	81.28%
3	71.98%	76.20%
4	80.85%	73.85%
5	78.55%	67.35%
overall	**77.68%**	**77.17%**

5 Conclusions and Future Work

In this paper, we have introduced a new version of the Czech benchmark dataset called Simple Question Answering database (SQAD). The new version 3.0 contains almost 13,500 records. In addition to new content, a new AddQA online tool for annotations of new SQAD records was presented.

In the future work, the AddQA online annotation tool is planned to incorporate all the consistency restrictions described in Section 3.

We have also presented the first results of the document selection and answer selection modules with the SQADv3 dataset reaching the best results of 78.92% mean access precision (MAP) and 85.95 mean reciprocal rank (MRR).

Acknowledgements This work has been partly supported by the Czech Science Foundation under the project GA18-23891S.

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. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010, pp. 177–186. Springer (2010)
2. Horák, A., Medved', M.: SQAD: Simple Question Answering Database. In: Eighth Workshop on Recent Advances in Slavonic Natural Language Processing, RASLAN 2014. pp. 121–128. Tribun EU, Brno (2014)
3. Lai, G., Xie, Q., Liu, H., Yang, Y., Hovy, E.: RACE: Large-scale ReAding Comprehension Dataset From Examinations. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 785–794 (2017)
4. Medved', M., Horák, A.: Sentence and word embedding employed in open question-answering. In: Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018). pp. 486–492. SCITEPRESS - Science and Technology Publications, Setúbal, Portugal (2018)
5. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text tokenisation using unitok. In: RASLAN 2014. pp. 71–75. Tribun EU, Brno, Czech Republic (2014)
6. Rajpurkar, P., Jia, R., Liang, P.: Know What You Don't Know: Unanswerable Questions for SQuAD. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 784–789. Associationfor Computational Linguistics, Melbourne, Australia (2018)
7. Šmerk, P.: Fast Morphological Analysis of Czech. In: Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2009. pp. 13–16 (2009)
8. Šulganová, T., Medved', M., Horák, A.: Enlargement of the Czech Question-Answering Dataset to SQAD v2.0. In: Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2017. pp. 79–84 (2017)
9. Šmerk, P.: K počítačové morfologické analýze češtiny (in Czech, Towards Computational Morphological Analysis of Czech). Ph.D. thesis, Faculty of Informatics, Masaryk University (2010)

Part IV

Text Corpora

SiLi Index: Data Structure for Fast Vector Space Searching

Ondřej Herman and Pavel Rychlý

Faculty of Informatics
Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xherman1,pary}@fi.muni.cz

Abstract. Nearest neighbor queries in high-dimensional spaces are expensive. In this article, we propose a method of building and querying a stand-alone data structure, SiLi (**Si**milarity **Li**st) Index, which supports approximating the results of k-NN queries in high-dimensional spaces, while using a significantly reduced amount of system memory and processor time compared to the usual brute-force search methods.

Keywords: word embeddings, vector space, semantic similarity

1 Introduction

1.1 Motivation

Vector space models have been central to the field of natural language processing for a long time, ranging from traditional sparse and high-dimensional bag-of-words document representations, where the vector space co-ordinates represent different words or phrases with tens of thousands of dimensions, to recent dense embeddings, which typically operate on hundreds of dimensions. The particular dimensions usually do not have clear interpretation, but as in the case of [2], the structure of the vector space has some interesting and useful properties.
Main operations of interest operating on dense vector spaces are the following:

1. **Pairwise similarity** – given two elements of the vector space, quantify their similarity.
2. ***k*-nearest neighbor queries** – given an element of the vector space, retrieve k most similar elements.
3. **Analogy queries** – given three elements, a, a^*, and b, retrieve k candidate elements b^* which satisfy the following criterion: a is to a^* as b is to b^*.

Evaluating pairwise similarity of two elements is cheap, as it is enough to retrieve the elements and then calculate the similarity. When the elements are stored in secondary storage, this means two seek operations and a single evaluation of similarity.

k-nearest neighbor query is significantly more demanding. The typical and widely deployed naïve algorithm calculates the pairwise similarity between the

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 111–116, 2019.

query vector and every other element of the vector space and then selects the top k most similar elements. Processor performance is usually not the limiting factor in this case. The vectors to be compared need to be loaded from storage. Even the cost of transferring the model to the processor has significant cost.

For example, a fastText model calculated for the word attribute of the enTenTen 2013 corpus [3] has over 6 million distinct elements, which represent all corpus lexicon entries, which occur in the corpus at least five times. The sizes of the model for different vector lengths are:

Dimension	Datatype	Total size	Note
100	float32	2.540 GiB	
300	float32	7.620 GiB	
500	float32	12.700 GiB	
500	float16	6.350 GiB	non-native datatype, slow

A recent desktop computer (as of 2019) has approximately 50 GiB/s of available bandwidth between the processor and main memory, so even when the model of dimension 100 is used, the rate at which the queries can be evaluated is limited to 20 per second by memory transfers alone.

The performance of evaluating analogy queries is comparable, as an analogy query can be transformed to the k-nearest neighbor query in the following way: given the query a is to a^* as b is to b^* where b^* is the vector we are looking for, a new vector v is calculated as $v = b - a + a*$ and then a k-nearest neighbor query around v is evaluated.

As can be seen, to obtain reasonable performance, the vector space elements to be searched need to be stored in main system memory – streaming the results from secondary storage would mean slowdown of two to three orders of magnitude.

2 Description

The main idea, on which the SiLi Index is based, is that it is not necessary to store the elements of the vector space themselves, but only the results of the k-nearest neighbor queries for every element.

2.1 Structure

The current version of the structure consists of three parts: the main record array, which stores nearest neighbors for every vector, an index, which provides mapping from numerical IDs to record array positions to enable fast lookups, and an external lexicon, which is a mapping between the lexicon elements and their IDs.

The lexicon is stored as a text file, with two lines of metadata. The first line describes the amount of elements contained in the model, while the second line is the dimensionality of the original vector space. The rest of the lines in the file are strings – lexicon elements, ordered by the frequency at which they appear in

the source corpus, with the IDs being implicitly encoded as the position of the lexicon element in the file:

6658558
100
the
.
,
to
and
of
a
in
is
that
⋮

The main data file consists of variable length records. Every record has a header, which contains two 4 byte values: the ID of the vector space element which is represented by the current vector, and the number n of the most similar elements stored in the record. Then follow n pairs of 4 byte integers, representing the n most similar elements and their similarities, ordered by the similarity in descending order. The maximum similarity is represented as the value of 2^{20}, while the minimum would be 0. For example, the record for the first element in the model build from enTenTen13 [3] would be:

Address	ID	Number of elements
	Neighbor ID	Similarity
0x000000	0	500
0x000008	0	1048575
0x000010	5	861673
0x000018	7	807157
0x000020	18	755537
0x000028	92	750653
0x000030	24	746858
0x000038	196	736494
0x000040	52	735677
0x000048	1	723016
	⋮	⋮

The third part of the structure maps the lexicon elements to the positions in the record array. The mapping from the lexicon IDs to positions is a flat binary table, where every element is an 8 byte long offset into the main data file. This way, only a single random access to the file is needed to locate the position in the main data file. The portion of this mapping corresponding to the portion of the lexicon as shown above would be:

Address	Offset
0x000000	0
0x000008	501
0x000010	1002
0x000018	1503
0x000020	2004
0x000028	2505
0x000030	3006
0x000038	3507
0x000040	4008
0x000048	4509
⋮	⋮

2.2 Generation

The SiLi Index consists of the most most similar elements for every word. Therefore, efficient calculation of similarities between all pairs of elements is essential. Common similarity measure is cosine similarity. If the vector space elements are normalized, this task reduces to matrix multiplication. The whole result is, however, very big, and would not fit in main memory. To work around this, we calculate similarities in batches, between a 100-element band and every element in the vector space. The resulting SiLi index for the enTenTen13 corpus consisting of 6658558 distinct elements was calculated in 75 m 38.690 s of wall-clock time.

2.3 *k*-nearest neighbor queries

Retrieving k neighbors is trivial. First, the query string is translated to the ID using the lexicon file. Then, the corresponding offset is located in the offset table, which points to the location at which the actual record is stored in the main data file. If the lexicon is loaded in main memory, this means two seek operations, compared to the need to scan the whole model in the case of dense storage of word embeddings. The main memory requirements are much smaller.

2.4 Analogy Queries

Surprisingly, it is possible to evaluate analogy queries using only the information about nearest neighbors of the elements. The query a is to a^* as b is to b^*, where b is the element to be found, we need retrieve the list of nearest neighbours of a, a^* and b^*, then calculate the intersection of these lists and evaluate the resulting similarity – we optimize

$$\arg\max_{b^*}(sim(b^*, b - a + a^*))$$

To evaluate similarity between two vectors, we use cosine similarity, defined as

$$sim(u, v) = cos(u, v) = \frac{u \cdot v}{\|u\| \cdot \|v\|}$$

Following the work of Mikolov et al. ([5]), we normalize the embedding vectors, so the similarity measure can be simplified to $sim(u, v) = u \cdot v$. Using basic algebra, the first equation can then be transformed ([6,4]) to the form

$$\arg\max_{b^*}(b^* \cdot b - b^* \cdot a + b^* \cdot a^*)$$

We store the cosine similarities of normalized vectors in the SiLi index, therefore $b^* \cdot b$, $b^* \cdot a$ and $b^* \cdot a^*$ can be extracted from the records for the elements b, a and a^* respectively. This way, many analogy queries can be evaluated, alleviating the need for storage of the complete embeddings.

2.5 Future Work

Currently, we store 500 nearest neighbors for every element in the vector space. This is likely unnecessary. Storing an adaptive amount of neighbors depending on the frequency of the word and its neighbor would likely result only in marginal loss of recall and accuracy of the model. Another interesting signal is the specific distribution of the neighbors of a specific word. However, we found that the distribution itself is not correlated with frequency of the lexicon elements (1), so more research on this topic would be necessary.

The headers in the main record array are not necessary, but currently are kept for completeness, as the record array can be used without the mapping table, or the mapping table could be recovered from it. Another source of inefficiency is the storage of the similarity in 4 bytes, even though the amount of useful information contained in this value is significantly lower. However, the current format is aligned and easily machine readable. The influence of different encodings with respect to performance of the model needs to be evaluated.

Perhaps the most significant improvement would be obtained by creating a new record type for rare lexicon elements, where only a very small amount of the representants would be stored, between 1 to 5. The queries would be carried through these representants. The result would not be exact for the low frequency lexicon elements anymore, but would yield result with lower and upper bounds on the similarity. Rare words usually do not have high-quality embeddings, so this might not cause significant issues. The drawback of this approach is however reduced performance, as more seeks to the underlying storage would need to be carried out.

Some other approaches employing locally sensitive hashing show promise, but we find them overly complex, such as the FALCONN library ([1]).

3 Conclusion

We presented a data structure, SiLi Index, and accompanying algorithms which enable efficient k-nearest neighbor query evaluation from data stored on secondary storage. The data structure can also support many important instances of analogy queries.

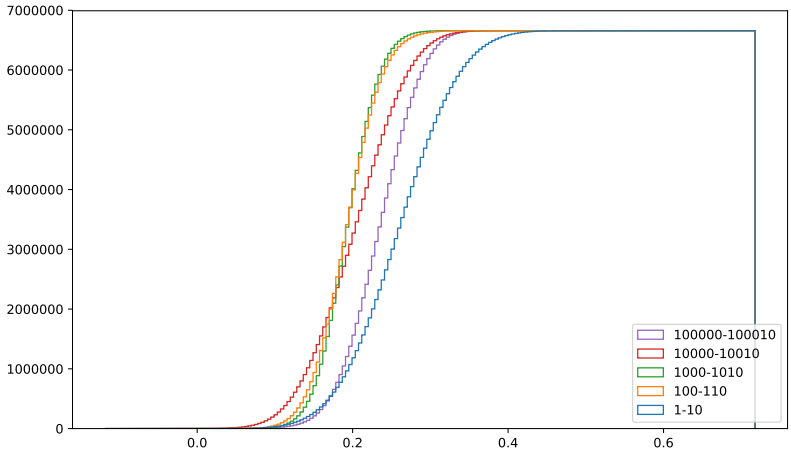


Fig. 1: Cumulative distribution of vector similarities for different rank bands for the word embedding model calculated using fastText on the enTenTen13 corpus.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin infrastructure LM2015071. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015.

References

1. Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., Schmidt, L.: Practical and optimal lsh for angular distance. In: Advances in Neural Information Processing Systems. pp. 1225–1233 (2015)
2. Grave, E., Mikolov, T., Joulin, A., Bojanowski, P.: Bag of tricks for efficient text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL. pp. 3–7 (2017)
3. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlỳ, P., Suchomel, V.: The tenten corpus family. In: Proceedings of the 7th International Corpus Linguistics Conference CL. pp. 125–127 (2013)
4. Levy, O., Goldberg, Y.: Linguistic regularities in sparse and explicit word representations. In: Proceedings of the eighteenth conference on computational natural language learning. pp. 171–180 (2014)
5. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
6. Rychlỳ, P.: Evaluation of the sketch engine thesaurus on analogy queries. RASLAN 2016 Recent Advances in Slavonic Natural Language Processing p. 147 (2016)

Quo Vadis, Math Information Retrieval

Petr Sojka, Vít Novotný, Eniafe Festus Ayetiran, Dávid Lupták, and Michal Štefánik

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
sojka@fi.muni.cz, {witiko,ayetiran,dluptak,stefanik.m}@mail.muni.cz
https://mir.fi.muni.cz/

Abstract. With the exponential growth of information in the digital form, information retrieval and querying digital libraries is of paramount importance, and mathematical and technical STEM documents are not an exception. The key for precise searching is the adequate and unambiguous representation of documents, paragraphs, sentences and words, which we are going to evaluate. We are presenting a roadmap to tackle the problem of searching and question answering in the digital mathematical libraries, and discuss the pros and cons of promising approaches primarily for the key part, namely the document representation: several types of embeddings, topic mixtures and LSTM. The listed representation learning options will be evaluated at the next ARQMath evaluation lab of CLEF 2020 conference.

Keywords: math information retrieval, question answering, STEM, digital mathematical libraries, embeddings, MIaS, MIaSNG, DML

"If not now, when?"
Chapters of the Fathers (Pirkei Avot, 1:14)

1 Introduction

Content is king. Content expressed in math formalism and formulae are often crucial and non-negligible part of the content of science, technology, engineering, and mathematics (STEM) papers. Mathematical formulae and diagrams, mostly due to their inherent structure and complexity, used to be not taken into account when processing the language of documents. Mathematical discourse, as a niche market, was not supported by tools for indexing and searching, digitization, or question answering.

There is already more than a decade of attempts to tackle the problem of Math Information Retrieval (MIR) in the Digital Mathematical Libraries (DML):

Infty, 2003 first math optical character recognition (OCR) system. [45]
MathDex, 2006 first search engine ever indexing MathML. [26]
DML-CZ, 2005–2009 one of the first attempts to classify and categorize mathematical knowledge [43] by automated means and tools: Gensim library has been designed for the DML usage [34,35].

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 117–128, 2019.

DML workshop series, 2008–2011 first workshop series specifically targeted for MIR and related problems.

EuDML, 2010–2013 first DML that deployed the MIaS search engine [41,42,52] designed for searching math formulae, and developed tools like WebMIaS [20] specifically for the STEM domain.

NTCIR 10, 2013 first evaluation competition with a MIR task. [1]

Tangent, 2012–2014 first visual-based indexing (symbol layout tree) and search engine, that is able to find any two dimensional mathematical and diagrammatical structures. [28,38]

MCAT, Aizawa MIR group, 2016 first system to learn how to effectively combine text and math for reranking. [14]

Equation embeddings, 2018 first joint embedding model that represents formulae by taking into account surrounding texts. [15]

Tangent CFT, 2019 first formula embedding model that uses two hierarchical representations: Symbol Layout Trees (SLTs) for appearance, and Operator Trees (OPTs) for mathematical content. [21]

TopicEq, 2019 first topic model that jointly generates mathematical equations and their surrounding text (TopicEq). [53]

ARQMath lab, c/o CLEF 2020 first answer retrieval task for questions on math data[1].

The accelerated pace with which the research in MIR continues, and new machine-learning approaches that appeared recently show promise of a new generation of search engines. It will take into account not only similar words or phrases, but the disambiguated meaning of structured objects like equations, sentences, paragraphs, or trains of thoughts.

We are going to develop the Math Indexer and Searcher of New Generation (MIaSNG) that will take into account the latest state of the art in the area of document meaning representation. To this end, we start by studying and evaluating several approaches in document representations.

The structure of the paper is as follows. In Section 2, we present recent approaches based on embeddings, specifically those that are capable of representing structured objects such as equations. In Section 3, a method based on joint text and math topic modeling is discussed. Another possibility of representing sequences of tree structures is described in Section 4. Section 5 evaluates versions of transfer learning for our goals. This list of possibilities is by no means exhaustive, but presents approaches for evaluation during the preparation of MIaSNG.

[1] `https://www.cs.rit.edu/~dprl/ARQMath/`

"The day is short, the labor vast, the toilers idle, the reward great,
and the Master of the house is insistent."
Chapters of the Fathers (Pirkei Avot, 2:20)

2 Joint Embeddings for Text and Math: Equation Embeddings

Since the seminal work of Mikolov et al. [25], unsupervised word embeddings have become the preferred word representations for many natural language processing tasks. Document similarity measures extracted from unsupervised word embeddings, such as the Soft Cosine Measure (SCM) [39], are fast [27] and achieve strong performance on semantic text similarity [5], textual information retrieval [9], and entrance exam question answering [39] tasks.

In mathematical discourse, formulae are often more important than words for understanding. [16] Unlike words, formulae are deeply structured and most are unique. Unlike unsupervised embeddings of words, which are many and well-understood, unsupervised embeddings of mathematical formulae are only now beginning to be explored. The SCM with joint embeddings of words and formulae can form a basis of a fast and accurate mathematical search engine.

2.1 EqEmb and EqEmb-U Models

Krstovski and Blei [15] propose unsupervised joint embeddings of math and formulae: EqEmb and EqEmb-U. Their approach is based on a) symbol layout tree (SLT) visual encoding of mathematical formulae from Zanibbi et al. [54], and b) unsupervised embeddings of Koopman-Darmois family probability distributions [37] that generalize Skipgram with negative sampling of Mikolov et al. [24].

In the EqEmb model, every formula is represented as a single vocabulary entry and its internal structure is disregarded. For every input word i, the EqEmb model predicts the context words and formulae, and for every input formula m, the EqEmb model predicts the context words.

In the EqEmb-U model, formulae are tokenized using the SLT visual encoding. For every input word i, the EqEmb-U model predicts the context words and formula tokens, and for every input formula token m, the EqEmb-U model predicts the context formula tokens. After training, formula embeddings are obtained by averaging the embeddings of the formula tokens.

Unlike the FastText model of Bojanowski et al. [4], neither EqEmb nor EqEmb-U take subword information into account. Unlike EqEmb, EqEmb-U formula embeddings do not take context words into account. Unlike Mansouri et al. [21], both EqEmb and EqEmb-U use only a visual encoding of formulae (SLT) and not the Operator Tree (OPT), which encodes the meaning of formulae.

2.2 Evaluation

Krstovski and Blei compare EqEmb and EqEmb-U to existing word embeddings [18,25,29,37] using log-likelihood over arXiv articles from the NLP, IR, AI, and ML domains. They obtain best results using EqEmb-U closely followed by EqEmb across all domains. Qualitative evaluation shows that kNN on EqEmb formula embeddings can be used to recommend highly related words and formulae.

"He who acquires a good name, has acquired himself something indeed."
Chapters of the Fathers (Pirkei Avot, 2:8)

3 Joint Text and Math Topic Modelling

With the increasing number of documents and their archives available in our digital era, it becomes more difficult to find content that interests us. Search engines play a fundamental role in the area of information retrieval and help us find a set of documents based on given keywords. However, sometimes we might be out of keywords and want to explore similar materials related to the theme of other ones. Probabilistic topic modeling is designed for this purpose – a set of statistical methods that analyzes words in the texts and discovers topics based on their content.

3.1 TopicEq Model

In the context of STEM fields, the text itself is not the only part of papers that delivers the message. Mathematics is ubiquitous, and it comprehensively communicates the ideas. However, most works in natural language processing or machine learning study text and math separately. Yasunuga et al. [53] reason that these two components should be studied jointly together, as the text surrounding the mathematical equations can provide context for its better understanding and vice versa. They propose a new model called TopicEq that applies a topic model to the text context and jointly the same latent topic proportion vector to generate a sequence of math symbols in a recurrent neural network (RNN).

In their work, they apply neural variational inference technique [22,23,44] to train topic models. They employ an RNN to model equations as a sequence of LaTeX tokens [13]. They relate to and extend Latent Dirichlet allocation (LDA) [3] as they model two different modalities – word text and math equations. They also demonstrate that RNN-based models [7] are more effective than the bag of token-based models for equation processing. And finally, this work relates to equation embeddings [15] with additional modeling of each equation as a sequence of symbols.

3.2 Implementation and Evaluation

The correlated topic model [2] that uses a log-normal distribution is a baseline for the TopicEq model. The new model introduces the surrounding text of an equation as a context, and the generative process takes the same latent topic proportion vector for both this context and the math expression. An RNN generates equations as a sequence of mathematical symbols from the vocabulary of LaTeX tokens and the extension of the LSTM [11], named Topic-Embedded LSTM (TE-LSTM), embeds the proportional vector inside the LSTM cell to keep the topic knowledge.

They perform experiments on the dataset of context-equation pairs, constructed from sampled 100,000 arXiv articles. They define a context as five

consecutive sentences both before and after the equation and kept equations only of a specific length, which yields the final 400,000 context-equation pairs. In the topic model evaluation, the results show that using the joint RNN equation model significantly improves the coherence of topics of scientific texts. In the equation model evaluation, TE-LSTM outperforms generic LSTM in reducing the perplexity and syntax error rate while also requiring less training time.

Qualitative analysis of the newly designed model shows its high capability to interconnecting the mathematics with the topics in several applications. In topic-aware equation generation, the generated equations reflect the characteristics of given topics, even if a mixture of topics is in question. In the equation topic inference, the TopicEq model performs better in precision and consistency than the bag-of-token baseline, and the topic-dependent alignment between mathematical tokens and words can predict results adequately based on the given topic.

TopicEq model can increase the interpretability of equations regarding their context and improve the exploring similar scientific documents. More experiments should be undergone for the very short as well as complex formulae to see the performance of this topic model. The selection of the proper word context length could also be crucial for the results.

"What is the right path a man should choose?
Whatever is honorable to himself, and honorable in the eyes of others."
Chapters of the Fathers (Pirkei Avot, 2:1)

4 Mathematical Expressions Embedding Using Tree-Structured Bidirectional LSTM

The past few years have witnessed an upsurge in the use of deep learning architectures for semantic representation of sequential data due to their ability to capture long-range dependencies. Long short-term memory networks [11] addresses the problem of exploding or vanishing gradients, which had hitherto made traditional recurrent neural networks (RNNs) unable to capture long-range correlations in a sequence such as text. The long-range dependencies are preserved with a memory cell.

The bi-directional LSTM [10] is a variant of the traditional LSTM, which consists of two LSTMs that are run simultaneously, one for the input sequence and the other for the reverse of the input. This is to enable the network model to learn in both directions, taking into account the left and the right contexts i.e. the past and the future information using the hidden state of the LSTM at each time step.

The bi-directional LSTM architecture is suitable for strictly sequential information propagation and cannot handle information with hierarchical structure in mathematical formulae and expressions. The tree-structured LSTMs [17,46,55,56] have been introduced to model the syntactic structure of natural language text but they have not yet been applied to mathematics.

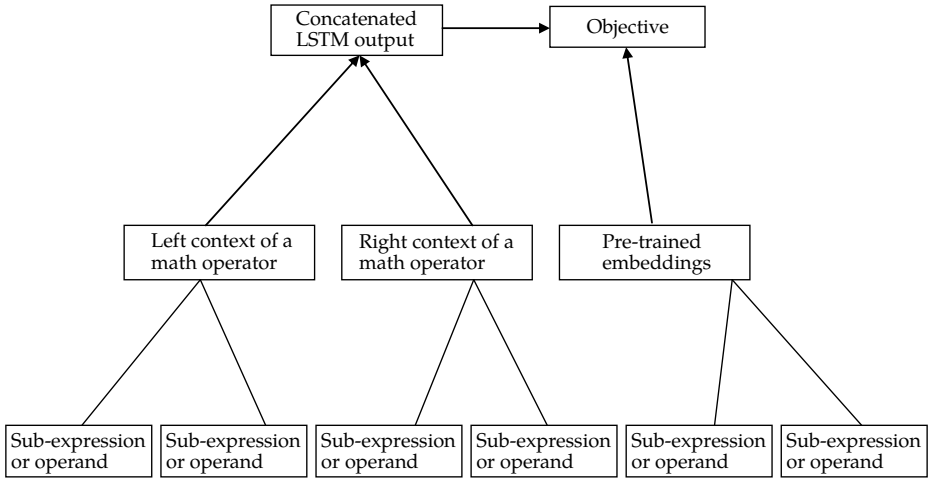


Fig. 1: Proposed Tree-Structured Bi-directional LSTM for Math Expressions Embedding

4.1 Proposed Bi-directional Tree-Structured LSTM

The general architecture of our proposed Bi-directional Tree-Structured LSTM is presented in Figure 1.

Recently, Thanda et al. [48], in their math information retrieval system for NTCIR-12 MathIR task used a bag-of-words version of Paragraph Vector (PV-DBOW) [18]. In their work, the math formulae in Wikipedia and arXiv papers are utilized. The math formulae are represented in the form of a tree, where the non-leaf nodes correspond to operators and the leaf nodes correspond to operands.

Our proposed method aims to use a similar approach but using the tree-structured bi-directional LSTM that can capture long-range dependencies. To represent any mathematical expression in a document, each mathematical operator will serve as the target token, and the two LSTMs will learn contexts in both directions. The operands will serve as the leaves, taking into account the structure of the formulae. First, the formulae or expressions will be parsed using ANTLR[2] or BISON[3] to recover their structure. The raw expressions will be converted to XML and MathML using the LaTeXML, the result of which will be canonicalized [40] and used to train a model using an adaptation of the Bi-directional Tree-structured LSTM [47] with a pre-trained embedding, using Word2Vec [24] as the objective.

As applicable to natural language texts, we hypothesize that the LSTM, apart from its capability to capture the long-range coherence in Math expressions will also be able to capture the order of combination of operators and operands alike,

[2] `https://www.antlr.org/`

[3] `https://www.gnu.org/software/bison/manual/bison.html`

taking into account notational variations. However, we envisage an outcome which may not be exact as the results reported for natural language texts due to the peculiarity of Math texts.

"In a place where there are no worthy men, strive to be worthy."
Chapters of the Fathers (Pirkei Avot, 2:5)

5 Representation and Transfer Learning for Math

Lately, we have been the observers of the dramatic movement of the reached quality in solving some of the principal high-level NLP Tasks [6,33,50]. Some of the new approaches have, in fact, overreached not only the current state-of-the-art by even tens of percent [8,51], but also the measured human performance [33].

The new technologies are based on a distinct set of ideas which has driven the development of their architecture. ELMo [30] builds upon a character-level convolution and a joint optimization of sequential language models (bi-LM), while attention-based transformer architectures, such as GPT [32] or BERT [8], utilize forward, or bi-directional incremental pooling of so-called attention [49] in forward, or bi-directional manner, respectively.

Yet, there is an attribute that intersects this new stream of methods. It is a fact they are pre-trained without a supervisor on a vast amount of data in the form of general language corpus [8,30,32]. Subsequently, they can be fine-tuned on downstream tasks [30], or their internal representations can be even used in zero-shot manner [32]. Furthermore, the generalization properties of transformers in language modeling [32] has even been underscored by their performance on language-agnostic downstream tasks [31], where some multilingual models are documented to perform well on a zero-shot classification of previously-unseen languages, even on ones that do not share any vocabulary with the fine-tuned language [31].

The surprising generalization capabilities of the attention-based technologies motivate us to evaluate their performance in the context of relevant tasks of math understanding. For our objectives, we propose several adaptations of the Transformer architecture, which reaches state-of-the-art results on other tasks, e.g. a question answering task [33] with an objective of explaining math formula variables, based on the surrounding context. Here, the variable denotation can be interpreted as a question, and the figure context as the answering paragraph. Similarly, we propose the use of sequence classification architecture used for paraphrase detection for the MIR formula-based search [20]: the similarity of their contexts can determine the disambiguation of the parts of the formulae.

We believe that the mentioned math understanding tasks could be fine-tuned from the weights pre-trained on general language modeling tasks [19], just like their native paired tasks. Subsequently, in cases where we can directly interpret our tasks in a framework of the native ones (SQuAD and MRPC, in the mentioned cases), we might be able to utilize the rich data sets of these native tasks to additionally fine-tune the pre-trained models in favour of our objectives.

Perhaps the main drawback of the methods above is their computational complexity. In any area related to the information retrieval, it is crucially

important to be able to either compute the representations of the documents on-the-fly, or to pre-index them and compare pairwise to the query in the real time.

It remains an open question whether the transferability of the models can also be successfully utilized in MIR. There are reports that the methods can be used to directly infer the context-dependent embeddings [8] of the input, or the internal representations can be fine-tuned to provide the similar embeddings of related paragraph pairs [36].

We believe that all the aforementioned methods are not necessarily bound to natural language applications: the successful inter-lingual applications [31] suggest that as long as the meaning of mathematical documents can be captured both in the natural language, just as in the math formulae, their general representation can be modeled.

We plan to design the experiments to evaluate whether the bidirectional sequence embeddings [12] or Transformers' internal representation [49] can be adapted to a general math representation. It is possible that the meaning shared among the math formulae and its interpretation in a natural text is still too latent to follow using a single model. Another eventual threat is that joint representations will, in comparison to standard, separate representations, miss some important low-level (e.g. morphological) features that are, however, crucial for relevant math information retrieval.

> "It is not incumbent upon you to complete the work,
> but neither are you at liberty to desist from it."
> Chapters of the Fathers (Pirkei Avot, 2:21)

6 Conclusion

This essay records the possibilities of representing the complex meaning of mathematical structures in STEM documents. Discussed representations reach state-of-the-art performance in text-only versions of NLP tasks, and their adaptation to cope with math seems feasible.

The paper thus serves as an outline of the prototypical implementation and evaluation of MIaSNG, the Math Indexing and Searching system of New Generation, which we are about to develop in the following years. We believe that having a joint representation of the meaning of both text and math will allow a new level of querying mathematical corpora such as arXiv or EuDML. Even though it seems that the Pareto principle holds – 80% of the conveyed message is in the form of text, the discussed approaches consistently show that coupling the text and math meaning increases the quality of language models, representations, and thus the future MIR via MIaSNG.

Acknowledgements This publication was written with the support of the Specific University Research provided by the Ministry of Education, Youth and Sports of the Czech Republic. 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. Aizawa, A., Kohlhase, M., Ounis, I.: NTCIR-10 Math Pilot Task Overview. In: Proc. of the 10th NTCIR Conference. pp. 654–661. NII, Tokyo, Japan (2013)
2. Blei, D.M., Lafferty, J.D.: A correlated topic model of science. The Annals of Applied Statistics **1**(1), 17–35 (06 2007). https://doi.org/10.1214/07-AOAS114
3. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet Allocation. The Journal of Machine Learning Research **3**, 993–1022 (2003)
4. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
5. Charlet, D., Damnati, G.: SimBow at SemEval-2017 task 3: Soft-cosine semantic similarity between questions for community question answering. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). pp. 315–319. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/S17-2051
6. Chen, S.F., Beeferman, D., Rosenfeld, R.: Evaluation Metrics for Language Models. In: DARPA Broadcast News Transcription and Understanding Workshop. pp. 275–280. CMU (1998)
7. Deng, Y., Kanervisto, A., Ling, J., Rush, A.M.: Image-to-markup Generation with Coarse-to-fine Attention. In: Proceedings of the 34th International Conference on Machine Learning – Volume 70. pp. 980–989. ICML'17, JMLR.org (2017), `http://dl.acm.org/citation.cfm?id=3305381.3305483`
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805 (2018), `https://arxiv.org/abs/1810.04805`
9. González Barbosa, J.J., Frausto-Solis, J., Villanueva, D., Valdés, G., Florencia, R., González, L., Mata, M.: Implementation of an Information Retrieval System Using the Soft Cosine Measure, vol. 667, pp. 757–766. Springer (12 2017). https://doi.org/10.1007/978-3-319-47054-2_50
10. Graves, A., Jaitly, N., Mohamed, A.: Hybrid speech recognition with deep bidirectional LSTM. In: Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU). pp. 273–278. Olomouc, Czech Republic (2013)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (Nov 1997). https://doi.org/10.1162/neco.1997.9.8.1735
12. Józefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., Wu, Y.: Exploring the Limits of Language Modeling. arXiv preprint arXiv:1602.02410 (2016), `https://arxiv.org/abs/1602.02410`
13. Karpathy, A.: The Unreasonable Effectiveness of Recurrent Neural Networks, `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`, Andrej Karpathy blog
14. Kristianto, G.Y., Topić, G., Aizawa, A.: Combining effectively math expressions and textual keywords in math IR. In: Proceedings of the 3rd International Workshop on Digitization and E-Inclusion in Mathematics and Science 2016 (DEIMS2016). pp. 25–32. sAccessNet (Nonprofit organization), Japan (2016), `http://workshop.sciaccess.net/DEIMS2016/articles/p02_Kristianto&Aizawa.pdf`
15. Krstovski, K., Blei, D.M.: Equation embeddings. arXiv preprint (2018), `https://arxiv.org/abs/1803.09123`
16. Larson, R.R., Reynolds, C., Gey, F.C.: The abject failure of keyword IR for mathematics search: Berkeley at NTCIR-10 math. In: Kando, N., Kato, T. (eds.) Proceedings of the

10th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-10, National Center of Sciences, Tokyo, Japan, June 18-21, 2013. National Institute of Informatics (NII) (2013)
17. Le, P., Zuidema, W.: Compositional distributional semantics with long short term memory. In: Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics. pp. 10–19. Denver, Colorado, USA (2015)
18. Le, Q., Mikolov, T.: Distributed Representations of Sentences and Documents. In: Proceedings of International Conference on Machine Learning. pp. 1188–1196. Beijing, China (2014)
19. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L.S., Stoyanov, V.: RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692 (2019), https://arxiv.org/abs/1907.11692
20. Líška, M., Sojka, P., Růžička, M.: Math indexer and searcher web interface: Towards fulfillment of mathematicians' information needs. In: Watt, S.M., Davenport, J.H., Sexton, A.P., Sojka, P., Urban, J. (eds.) Intelligent Computer Mathematics CICM 2014. Proceedings of Calculemus, DML, MKM, and Systems and Projects. pp. 444–448. Springer International Publishing Switzerland, Zurich (2014). https://doi.org/10.1007/978-3-319-08434-3_36, https://arxiv.org/abs/1404.6476
21. Mansouri, B., Rohatgi, S., Oard, D.W., Wu, J., Giles, C.L., Zanibbi, R.: Tangent-CFT: An Embedding Model for Mathematical Formulas. In: Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval. pp. 11–18. ICTIR '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3341981.3344235
22. Miao, Y., Grefenstette, E., Blunsom, P.: Discovering discrete latent topics with neural variational inference. In: Proceedings of the 34th International Conference on Machine Learning – Volume 70. pp. 2410–2419. ICML'17, JMLR.org (2017), http://dl.acm.org/citation.cfm?id=3305890.3305930
23. Miao, Y., Yu, L., Blunsom, P.: Neural variational inference for text processing. In: International conference on machine learning. pp. 1727–1736. ICML'16, JMLR.org (2016), http://dl.acm.org/citation.cfm?id=3045390.3045573
24. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 26, pp. 3111–3119. Curran Associates, Inc. (2013)
25. Mikolov, T., Chen, K., et al.: Efficient estimation of word representations in vector space. arXiv preprint (2013), https://arxiv.org/abs/1301.3781v3, accessed 22 October 2019
26. Munavalli, R., Miner, R.: MathFind: a math-aware search engine. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 735–735. SIGIR '06, ACM, New York, NY, USA (2006). https://doi.org/10.1145/1148170.1148348
27. Novotný, V.: Implementation Notes for the Soft Cosine Measure. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. pp. 1639–1642. CIKM '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3269206.3269317
28. Pattaniyil, N., Zanibbi, R.: Combining TF-IDF Text Retrieval with an Inverted Index over Symbol Pairs in Math Expressions: The Tangent Math Search Engine at NTCIR 2014. In: Kando, N., Joho, H., Kishida, K. (eds.) Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies. pp. 135–142. National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430 Japan, Tokyo (2014), http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings11/pdf/NTCIR/Math-2/08-NTCIR11-MATH-PattaniyilN.pdf

29. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
30. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the ACL: Human Language Technologies, Volume 1 (Long Papers). pp. 2227–2237. Association for Computational Linguistics, New Orleans, Louisiana (Jun 2018). https://doi.org/10.18653/v1/N18-1202
31. Pires, T., Schlinger, E., Garrette, D.: How multilingual is Multilingual BERT? (2019)
32. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog **1**(8) (2019)
33. Rajpurkar, P., Jia, R., Liang, P.: Know what you don't know: Unanswerable questions for SQuAD. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 784–789. Association for Computational Linguistics, Melbourne, Australia (Jul 2018). https://doi.org/10.18653/v1/P18-2124, `https://www.aclweb.org/anthology/P18-2124`
34. Řehůřek, R., Sojka, P.: Automated Classification and Categorization of Mathematical Knowledge. In: Autexier, S., Campbell, J., Rubio, J., Sorge, V., Suzuki, M., Wiedijk, F. (eds.) Intelligent Computer Mathematics—Proceedings of 7th International Conference on Mathematical Knowledge Management MKM 2008. Lecture Notes in Computer Science LNCS/LNAI, vol. 5144, pp. 543–557. Springer-Verlag, Berlin, Heidelberg (Jul 2008)
35. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010). https://doi.org/10.13140/2.1.2393.1847, `http://is.muni.cz/publication/884893/en`, software available at `http://nlp.fi.muni.cz/projekty/gensim`
36. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks (2019)
37. Rudolph, M., Ruiz, F., Mandt, S., Blei, D.: Exponential family embeddings. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 478–486. Curran Associates, Inc. (2016), `http://papers.nips.cc/paper/6571-exponential-family-embeddings.pdf`
38. Schellenberg, T., Yuan, B., Zanibbi, R.: Layout-based substitution tree indexing and retrieval for mathematical expressions. In: Viard-Gaudin, C., Zanibbi, R. (eds.) Document Recognition and Retrieval XIX. vol. 8297, pp. 126–133. International Society for Optics and Photonics, SPIE (2012). https://doi.org/10.1117/12.912502
39. Sidorov, G., et al.: Soft similarity and soft cosine measure: Similarity of features in vector space model. CyS **18**(3), 491–504 (2014). https://doi.org/10.13053/cys-18-3-2043
40. Sojka, P.: Exploiting semantic annotations in math information retrieval. In: Kamps, J., Karlgren, J., Mika, P., Murdock, V. (eds.) Proceedings of ESAIR 2012 c/o CIKM 2012. pp. 15–16. Association for Computing Machinery, Maui, Hawaii, USA (2012). https://doi.org/10.1145/2390148.2390157
41. Sojka, P., Lee, M., Řehůřek, R., Hatlapatka, R., Kucbel, M., Bouche, T., Goutorbe, C., Anghelache, R., Wojchiechowski, K.: Toolset for Entity and Semantic Associations – Final Release (Feb 2013), deliverable D8.4 of EU CIP-ICT-PSP project 250503 EuDML: The European Digital Mathematics Library
42. Sojka, P., Líška, M.: The Art of Mathematics Retrieval. In: Proceedings of the ACM Conference on Document Engineering, DocEng 2011. pp. 57–60.

Association of Computing Machinery, Mountain View, CA, USA (Sep 2011), https://doi.org/10.1145/2034691.2034703
43. Sojka, P., Řehůřek, R.: Classification of Multilingual Mathematical Papers in DML-CZ. In: Sojka, P., Horák, A. (eds.) Proceedings of Recent Advances in Slavonic Natural Language Processing—RASLAN 2007. pp. 89–96. Masaryk University, Karlova Studánka, Czech Republic (Dec 2007)
44. Srivastava, A., Sutton, C.A.: Autoencoding Variational Inference For Topic Models. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings. OpenReview.net (2017), https://openreview.net/forum?id=BybtVK9lg
45. Suzuki, M., Tamari, F., Fukuda, R., Uchida, S., Kanahori, T.: INFTY — An Integrated OCR System for Mathematical Documents. In: Vanoirbeek, C., Roisin, C., Munson, E. (eds.) Proc. of ACM Symposium on Document Engineering 2003. pp. 95–104. ACM, Grenoble, France (2003)
46. Tai, K.S., Socher, R., Manning, C.D.: Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. pp. 1556–1566. Beijing, China (2015)
47. Teng, Z., Zhang, Y.: Head-lexicalized bidirectional tree LSTMs. Transactions of the Association for Computational Linguistics **5**, 163–177 (2017). https://doi.org/10.1162/tacl_a_00053
48. Thanda, A., Agarwal, A., Singla, K., Prakash, A., Gupta, A.: A Document Retrieval System for Math Queries. In: Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies. pp. 346–353. Tokyo, Japan (2016)
49. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is All you Need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 5998–6008. Curran Associates, Inc. (2017), https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
50. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding (2018)
51. Warstadt, A., Singh, A., Bowman, S.R.: Neural Network Acceptability Judgments. arXiv preprint arXiv:1805.12471 (2018), https://arxiv.org/abs/1805.12471
52. Wojciechowski, K., Nowiński, A., Sojka, P., Líška, M.: The EuDML Search and Browsing Service – Final (Feb 2013), deliverable D5.3 of EU CIP-ICT-PSP project 250503 EuDML: The European Digital Mathematics Library, revision 1.2 https://project.eudml.eu/sites/default/files/D5_3_v1.2.pdf
53. Yasunaga, M., Lafferty, J.D.: TopicEq: A Joint Topic and Mathematical Equation Model for Scientific Texts. Proceedings of the AAAI Conference on Artificial Intelligence **33**, 7394–7401 (07 2019). https://doi.org/10.1609/aaai.v33i01.33017394
54. Zanibbi, R., Davila, K., Kane, A., Tompa, F.W.: Multi-stage math formula search: Using appearance-based similarity metrics at scale. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. pp. 145–154. ACM (2016)
55. Zhang, X., Lu, L., Lapata, M.: Top-down Tree Long Short-Term Memory Networks. In: Proceedings of NAACL-HLT. pp. 310–320. San Diego, California (2016)
56. Zhu, X., Sobhani, P., Guo, H.: Long short-term memory over recursive structures. In: Proceedings of the 32nd International Conference on Machine Learning. pp. 1604–1612. Lille, France (2015)

Discriminating Between Similar Languages Using Large Web Corpora

Vít Suchomel

Natural Language Processing Centre
Faculty of Informatics
Botanická 68a, Brno, Czech Republic
xsuchom2@fi.muni.cz

Lexical Computing
Brno, Czech Republic

Abstract. This paper presents a method for discriminating similar languages based on wordlists from large web corpora. The main benefits of the approach are language independency, a measure of confidence of the classification and an easy-to-maintain implementation.
The method is evaluated on VarDial 2014 workshop data set. The result accuracy is comparable to other methods successfully performing at the workshop.
A tool implementing the method in Python can be obtained from web site http://corpus.tools/.

Keywords: language identification, discriminating similar languages, building web corpora

1 Introduction

Language identification is a procedure necessary for building monolingual text corpora from the web. For obvious reasons, discriminating similar languages is the most difficult case to deal with. Continuing in the steps of our previous work [2], our goal in corpus building is to keep documents in target languages while removing texts in other, often similar languages. The aim is to process text of billion-word sized corpora using efficient and language independent algorithms. Precision (rather than recall), processing speed and easy-to-maintain software design are of key importance to us.

Data to evaluate language discrimination methods have been created by the organisers of the workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects (VarDial) since 2014 [10,11,7,9]. Various media ranging from nice newspaper articles to short social network texts full of tags were made available. Successful participants of this series of workshops have published their own approaches to the problem.

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 129–135, 2019.

2 Method

2.1 The Aim And Desired Properties

The aim of the method presented in this paper is to provide a simple and fast way to separate a large collection of documents from the web by language. This is the use case: Millions of web pages are downloaded from the web using a web crawler. To build monolingual corpora, one has to split the data by language.

Since the set of internet national top level domains (TLDs) targeted by the crawler is usually limited and a similarity of the downloaded texts to the target languages can be easily measured using e.g. a character n-gram model [6], one can expect only a limited set of languages similar to the target languages to discriminate. The method should work with both documents in languages that have been discerned in the past as well as texts in languages never processed before.

The presented method does:

- Enable supporting new languages easily (that implies the same way for adding any language).
- Allow adding a language never worked with before, using just the web pages downloaded or a resource available for all languages (e.g. articles from Wikipedia).
- Not use language specific resources varying for each language supported (e.g. a morphological database) – since that makes supporting new languages difficult.
- Apply to any structure of text, e.g. documents, paragraphs, sentences.
- Provide a way to measure the contribution of parts of a text, e.g. paragraphs, sentences, tokens, to the final classfication of the structure of the text.
- Provide a measure of confidence to allow setting a threshold and classfying documents below the threshold of minimal confidence as mixed or unknown language.
- Work fast even with collections of millions of documents.

2.2 Method Description

This method uses the initial step of the algorithm described in [2]. The reason for not including the expectation-maximisation steps is the aim to decrease the complexity of the solution, keeping the data processing time reasonably short.

The method exploits big monolingual collections of web pages downloaded in the past or even right before applying the method (i.e. using the text to identify its language as the method data source at the same time). The language of documents in such collections should be determined correctly in most cases, however some mistakes must be accepted since there are many foreign words in monolingual web corpora since e.g. foreign named entities or quotes are preserved. Even a lot of low frequency noise can be tolerable. Lists of words with relative frequency are built from these big monolingual collections of web

pages. The method uses the decimal logarithm of word count per billion words to determine the relative wordlist score of each word from the list of words accroding to the following formula:

$$score(w) = \log_{10}\left(\frac{f(w)\cdot 10^9}{|D|}\right)$$

Where $f(w)$ is the corpus frequency of the word (number of occurrences of the word in the collection) and $|D|$ is the corpus size (number of all occurrences of all words in the collection).

The wordlist is built for all languages to discern, prior to reading the input text. Usually, when building corpora from the web, languages similar to the target languages and languages prevalent in the region of the internet national top level domains occurring in the crawled data are considered. A big web corpus is a suitable source. To improve the list by reducing the presence of foreign words, limiting the national TLD of source web pages is advisable. E.g. using texts from TLD `.cz` to create a Czech word list should, intuitively, improve precision at a slight cost of recall.

The input of the method, i.e. the documets to separate by language, must be tokenised. Unitok [8] was used to tokenise text in all sources used in this work. Then, for each word in the input, the relative wordlist score is retrieved from each language wordlist. The scores of all words in a document grouped by the language are summed up to calculate the language score of a document. The same can be done for paragraphs or sentences or any corpus structure.

$$document\ score(language) = \sum_{w \in document} language\ score(w)$$

The language scores of a document are sorted and the ratio of two highest scoring languages is computed to determine the confidence of the classification. The score ratio is compared to a pre-set confidence threshold. If the ratio is below the threshold, the document is marked as a mixed language text and not included in the final collection of monolingual corpora. Otherwise the result language is the language with the highest score.

$$confidence\ ratio(document) = \frac{document\ score(top\ language)}{document\ score(second\ top\ language)}$$

According to our experience, setting the confidence threshold quite low (e.g. to 1.005) is advisable in the case of discerning very similar languages while higher values (e.g. 1.05) work for other cases (e.g. Czech vs. Slovak, Norwegian vs. Danish).

We usually understand a paragraph to be the largest structure consisting of a single language in the case of multilanguage web pages. The method presented in this work allows separating paragraphs in different languages found in a single multilingual document to multiple monolingual documents. Although code switching within a paragraph is possible, detecting that phenomenon is beyond the scope of this work.

The following sample shows the overall sentence language scores as well as particular word language scores in a sentence from VarDial 2014 test data. Words 'scheme', 'council' and 'tenant' contribute the most to correctly classifying the sample as British English (rather than American English). Column description: Word, en-GB score, en-US score. Punctuation was omitted from the wordlists thus getting a zero score.

```
<s lang="en-GB" confidence_ratio="1.018" en-GB="122.04" en-US="119.89">
Under     5.74    5.74
the       7.77    7.75
rent      4.70    4.59
deposit   4.56    4.40
bond      4.49    4.63
scheme    5.26    4.41
,         0.00    0.00
the       7.77    7.75
council   5.56    5.20
pays      4.20    4.26
the       7.77    7.75
deposit   4.56    4.40
for       7.06    7.07
a         7.36    7.34
tenant    4.34    3.94
so        6.34    6.31
they      6.51    6.50
can       6.53    6.54
rent      4.70    4.59
a         7.36    7.34
property  5.38    5.37
privately 4.05    3.99
.         0.00    0.00
</s>
```

3 Evaluation

The method was used to build language wordlists from sources described in the next subsection and evaluated on six groups of similar languages.

3.1 Wordlists

In this work, TenTen web corpus family [4] was used to build the language wordlists. Aranea web corpora [1] were used in addition to TenTen corpora in the case of Czech and Slovak. bsWaC, hrWaC and srWaC web corpora [5] were used in the case of Bosnian, Croatian and Serbian. All words, even hapax legomena were included in the wordlists. The source web pages were limited to the respective national TLD where possible.

Another set of wordlists to compare the method to other approaches was obtained from the DSL Corpus Collection[1] v. 1 made available at VarDial in 2014 [10].[2]

The last couple of wordlists for the purpose of evaluating the method was taken from corpus GloWbE comprising of 60 % blogs from various English speaking countries [3].[3]

The sizes and source TLDs of the wordlists are shown in Table 1. The difference of wordlist sizes is countered by using the relative counts in the algorithm.

Table 1: Sizes of wordlists used in the evaluation. Large web sources – TenTen, Aranea and WaC corpora – were limited to respective national TLDs. Other wordlists were built from the training and evaluation data of DSL Corpus Collection and parts of GloWbE corpus.

Language	Web TLD	Web wordlist	DSL wordlist	GloWbE wordlist
Bosnian	.ba	2,262,136	51,337	
Croatian	.hr	6,442,922	50,368	
Serbian	.rs	3,510,943	49,370	
Indonesian	–	860,827	48,824	
Malaysian	–	1,346,371	34,769	
Czech	.cz	26,534,728	109,635	
Slovak	.sk	5,333,581	121,550	
Portuguese, Brazilian	.br	9,298,711	52,612	
Portuguese, European	.pt	2,495,008	51,185	
Spanish, Argentine	.ar	6,376,369	52,179	
Spanish, Peninsular	.es	8,396,533	62,945	
English, Great Britain	.uk	6,738,021	42,516	1,222,292
English, United States	.us	2,814,873	42,358	1,245,821

3.2 Discriminating Similar Languaes – VarDial Workshop

The evaluation of the language separation method described in this paper on DSL Corpus Collection v. 1 gold data[4] performed by the original evaluation

[1] http://ttg.uni-saarland.de/resources/DSLCC/

[2] http://corporavm.uni-koeln.de/vardial/sharedtask.html

[3] http://www.corpusdata.org/

[4] https://bitbucket.org/alvations/dslsharedtask2014/src/master/test-gold.txt

script[5] can be found in Table 2. The result overall accuracy is compared to the best result presented at VarDial 2014[6]

Table 2: Overall accuracy using large web corpus wordlists and DSL CC v. 1 training data wordlists on DSL CC v. 1 gold data. The best result achieved by participants in VarDial 2014 can be found in the last column.

Languages	Wordlist	Accuracy	DSL Best
English GB/US	Large web corpora	**0.6913**	0.6394
English GB/US	GloWbE	**0.6956**	0.6394
English GB/US	DSL training data	0.4706	**0.6394**
Other languages	Large web corpora	0.8565	**0.8800**
Other languages	DSL training data	0.9354	**0.9571**
Bosnian, Croatian, Serbian	DSL training data	0.8883	**0.9360**
Indonesian, Malaysian	DSL training data	**0.9955**	**0.9955**
Czech, Slovak	DSL training data	**1.0000**	**1.0000**
Portuguese BR/PT	DSL training data	0.9345	**0.9560**
Spanish AR/ES	DSL training data	0.8820	**0.9095**

The wordlist based language separation method performed comparably to the results of participants of VarDial 2014.

DSL data wordlists might have perfomed better than large web corpora wordlists on the DSL test data since DSL training sentences were more similar to test sentences than web documents. The results show that large web corpus based wordlists performed better than the DSL test data based wordlists in the case of discerning British from American English.

4 Conclusion and Future Work

A Python script implementing the method presented in this paper can be found at `http://corpus.tools/` under name *Language Filter*. In our experience with Czech and Slovak, with Norwegian and Danish, and with filtering English or languages similar to the target language out of many monolingual web corpora, the quality of the result corpus greatly benefited from applying this simple yet powerful script.

We might consider including the expectation-maximisation steps decribed in the original algorithm [2] in a separate version of the script in the future to evaluate discriminating language variants of Spanish (Peninsular vs. American variants), Portuguese (European vs. Brazilian), French (Hexagonal vs. Canadian).

[5] `https://bitbucket.org/alvations/dslsharedtask2014/src/master/dslevalscript.py`

[6] `http://htmlpreview.github.io/?https://bitbucket.org/alvations/dslsharedtask2014/downloads/dsl-results.html`

Training and test data from more recent VarDial workshops will be used to evaluate the performance on additional language groups, such as Bulgarian/Macedonian, Hexagonal/Canadian French, or Persian/Dari.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin infrastructure LM2015071. This publication was written with the support of the Specific University Research provided by the Ministry of Education, Youth and Sports of the Czech Republic. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015.

References

1. Benko, V.: Aranea: Yet Another Family of (Comparable) Web Corpora. In: TSD. pp. 247–254
2. Herman, O., Suchomel, V., Baisa, V., Rychlý, P.: Dsl shared task 2016: Perfect is the enemy of good language discrimination through expectation–maximization and chunk-based language model. In: Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3). pp. 114–118 (2016)
3. Davies, M., Fuchs, R.: Expanding horizons in the study of World Englishes with the 1.9 billion word Global Web-based English Corpus (GloWbE). English World-Wide, 36(1), pp. 1–28. (2015)
4. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., Suchomel, V.: The tenten corpus family. In: 7th International Corpus Linguistics Conference CL. pp. 125–127 (2013)
5. Ljubešić, N., Klubička, F.: bs, hr, sr wac-web corpora of bosnian, croatian and serbian. In: Proceedings of the 9th Web as Corpus Workshop (WaC-9). pp. 29–35 (2014)
6. Lui, M., Baldwin, T.: langid.py: An off-the-shelf language identification tool. In: Proceedings of the ACL 2012 System Demonstrations. pp. 25–30. Association for Computational Linguistics, Jeju Island, Korea (Jul 2012), `https://www.aclweb.org/anthology/P12-3005`
7. Malmasi, S., Zampieri, M., Ljubešić, N., Nakov, P., Ali, A., Tiedemann, J.: Discriminating between similar languages and arabic dialect identification: A report on the third dsl shared task. In: Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3). pp. 1–14. Osaka, Japan (December 2016)
8. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text tokenisation using unitok. In: RASLAN. pp. 71–75 (2014)
9. Zampieri, M., Malmasi, S., Ljubešić, N., Nakov, P., Ali, A., Tiedemann, J., Scherrer, Y., Aepli, N.: Findings of the vardial evaluation campaign 2017. In: Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial). pp. 1–15. Association for Computational Linguistics, Valencia, Spain (April 2017)
10. Zampieri, M., Tan, L., Ljubešić, N., Tiedemann, J.: A report on the dsl shared task 2014. In: Proceedings of the first workshop on applying NLP tools to similar languages, varieties and dialects. pp. 58–67 (2014)
11. Zampieri, M., Tan, L., Ljubešić, N., Tiedemann, J., Nakov, P.: Overview of the dsl shared task 2015. In: Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects. pp. 1–9 (2015)

Evaluation of Czech Distributional Thesauri

Pavel Rychlý

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

Abstract. Distributional thesauri play a big role in current approaches to natural language applications. There are many ways how to build them and there is no clear way how to compare them which one is better. There are data sets for thesaurus quality evaluation for several languages, the ones for Czech are hard to use for several reasons. This paper proposes a new data set for Czech which is easy to use in different environments.

Keywords: distributional thesaurus, evaluation, outlier detection

1 Introduction

A thesaurus lists words with similar meaning for any given word. There is a long history of algorithms for automatic generation of distributional thesauri based on co-occurrence of words in a very large text. These algorithms find words occurring in same or similar contexts. That could include synonyms, antonyms (as one can expect in a human-created thesaurus) but also words from the same class (like animals) or hypernyms and hyponyms. With the availability of huge text collections, these data sets have good coverage of words and provide important information about words in a language. That is the reason they are successfully used in many natural language applications.

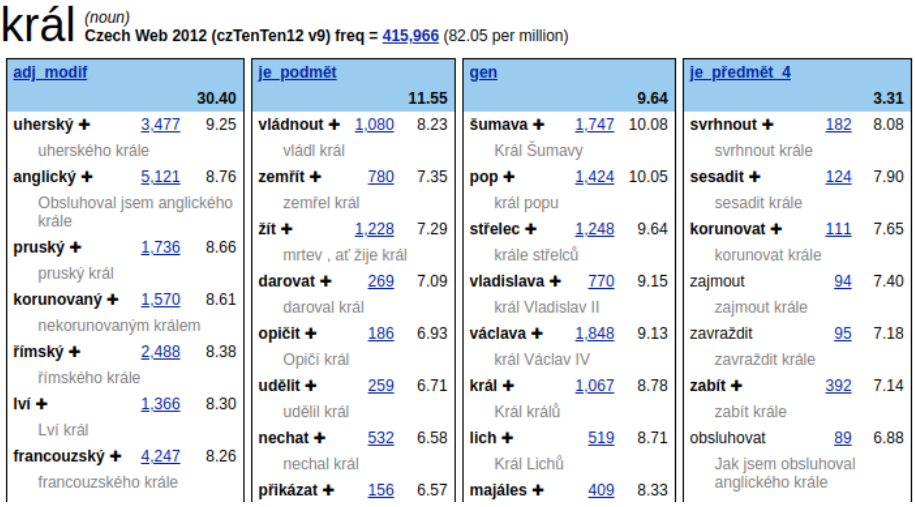
To compare which algorithm and/or settings of building a thesaurus is better there are methods for evaluating thesauri from the very beginning of building automatic thesauri in 1965 [7]. Thesaurus evaluation is discussed in more details in the next section.

The Sketch Engine (SkE) [4] is a corpus management system with several unique features. One of the most important feature (which also gave the name to the whole system) is a word sketch. It is a one page overview of grammatical and collocational behavior of a given word. It is an extension of the general collocation concept used in corpus linguistics in that they group collocations according to particular grammatical relation (e.g. subject, object, modifier etc.). The system is language independent and this paper will deal with Czech corpora and data generated by the system from them.

An example of word sketch for noun *král (king)* on CzTenTen12 [10] is in Figure 1. The Sketch Engine provides also a thesaurus. It is based on word sketches, similarity of two words is computed as the intersection of collocations

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 137–142, 2019.

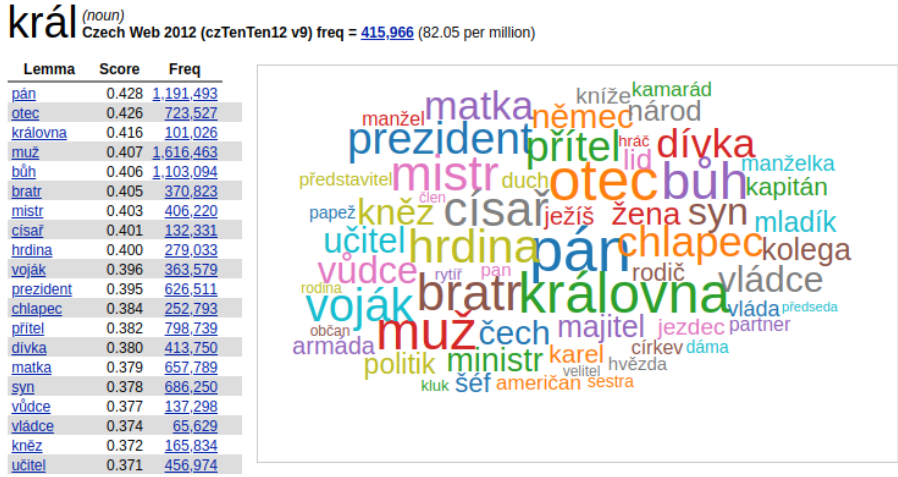
in respective grammatical relations of both words. An example of the thesaurus result for noun *král* is in Figure 2. More details about the algorithms behind the thesaurus computation can be find in [9].



král *(noun)*
Czech Web 2012 (czTenTen12 v9) freq = 415,966 (82.05 per million)

adj_modif			je_podmět			gen			je_předmět_4		
		30.40			11.55			9.64			3.31
uherský +	3,477	9.25	**vládnout +**	1,080	8.23	**šumava +**	1,747	10.08	**svrhnout +**	182	8.08
uherského krále			vládl král			Král Šumavy			svrhnout krále		
anglický +	5,121	8.76	**zemřít +**	780	7.35	**pop +**	1,424	10.05	**sesadit +**	124	7.90
Obsluhoval jsem anglického krále			zemřel král			král popu			sesadit krále		
pruský +	1,736	8.66	**žít +**	1,228	7.29	**střelec +**	1,248	9.64	**korunovat +**	111	7.65
pruský král			mrtev , ať žije král			krále střelců			korunovat krále		
korunovaný +	1,570	8.61	**darovat +**	269	7.09	**vladislava +**	770	9.15	zajmout	94	7.40
nekorunovaným králem			daroval král			král Vladislav II			zajmout krále		
římský +	2,488	8.38	**opičit +**	186	6.93	**václava +**	1,848	9.13	zavraždit	95	7.18
římského krále			Opičí král			král Václav IV			zavraždit krále		
lví +	1,366	8.30	**udělit +**	259	6.71	**král +**	1,067	8.78	**zabít +**	392	7.14
Lví král			udělil král			Král králů			zabít krále		
francouzský +	4,247	8.26	**nechat +**	532	6.58	**lich +**	519	8.71	obsluhovat	89	6.88
francouzského krále			nechal král			Král Lichů			Jak jsem obsluhoval anglického krále		
			přikázat +	156	6.57	**majáles +**	409	8.33			

Fig. 1: Word Sketch of word *král (king)* on CzTenTen12 corpus.



král *(noun)*
Czech Web 2012 (czTenTen12 v9) freq = 415,966 (82.05 per million)

Lemma	Score	Freq
pán	0.428	1,191,493
otec	0.426	723,527
královna	0.416	101,026
muž	0.407	1,616,463
bůh	0.406	1,103,094
bratr	0.405	370,823
mistr	0.403	406,220
císař	0.401	132,331
hrdina	0.400	279,033
voják	0.396	363,579
prezident	0.395	626,511
chlapec	0.384	252,793
přítel	0.382	798,739
dívka	0.380	413,750
matka	0.379	657,789
syn	0.378	686,250
vůdce	0.377	137,298
vládce	0.374	65,629
kněz	0.372	165,834
učitel	0.371	456,974

Fig. 2: Sketch Engine Thesaurus for word *král (king)* on CzTenTen12 corpus.

2 Thesaurus Evaluation

The first methods of evaluating thesaurus quality was based on gold standards – data prepared by several annotators. They contain a list of word pairs together with a numeric or quality assignment of their similarity. There are several problems with such data:

- some gold standards do not distinguish between similarity and relatedness (*money – bank*: score 8.5 out of 10 in WordSim353 data set [3])
- some gold standards do not provide any measure of similarity [6]
- usually, inter annotator agreement within human annotators is low.

Examples of different lists of similar words are discussed in [8].

The high attention to thesauri usage came from recent systems for computing vector representation of words [5], [1]. They bring a new methods of evaluating such representations. The most popular is the task of analogy queries. Each query is in form "*A* is to *B* as *C* is to *D*", where *D* is hidden and the system must guess it. The description of several variants of the task and the application of the task for the Sketch Engine thesaurus is in [8].

The biggest problem of this task is under-specification of a query. In many cases, humans are able to answer a question correctly because they know the query type. For example in the query *"Germany" is to "Berlin" as "France" is to "?"*, the right answer is *"Paris"* because the query is in the set *Capitals*. But Berlin could be the biggest city or the city where the respective president was born or anything else. These queries are also sensitive to the size of the training texts, they contains quite rare name entities. The evaluation therefore focuses on not so important parts of a language.

The most reliable evaluation seems to be so called outlier detection, proposed in [2]. The query in this evaluation set contains a list of words and the task is to find an outlier of that list, the word which is the least "similar" to the other words in the list. The outlier is computed from a thesaurus as a word for which the rest of the given list of words is the most compact. The compactness is defined as the sum (or average) of similarities of all word pairs in a cluster.

3 Evaluating Outlier Detection

The paper introducing this evaluation contains the whole data set. The outlier detection queries are defined in a form of two sets:

- positive set of 8 words forming a well understood cluster,
- negative set of 8 words with outliers.

It is possible to create 8 different queries from each such set pair. One word from the negative set and the whole positive set defines one such query. The article [2] contains 8 pairs of word sets. That mean it is possible to generate $8 \times 8 = 64$ individual queries.

The evaluation of a thesaurus computes two numbers:

- Accuracy – the percentage of successfully answered queries,
- Outlier Position Percentage (OPP) Score – average percentage of the right answer (Outlier Position) in the list of possible clusters ordered by their compactness.

Outlier Position (OP) is a number from 0 to the number of words in the query (9 in the original data set). 0 means the worst guess, the maximum means the right answer. Therefore OPP 100 % indicates all hits which means 100 % accuracy.

OOP provides more fine grained evaluation than accuracy. For incorrect answers it differentiates the position of the right answer in the ordered list.

There are several problems with the original data set for evaluating Czech thesauri. The crucial one is that it contains only English words. That is the main reason we have created our new data set. The second problem is that the evaluation uses the exact word forms in the queries. That is usually not a big problem for English because thesaurus is usually compiled for word forms. Also words in the queries are in the basic form in almost all cases. In Czech there is some times no obvious basic form. For example, many systems use masculine form of adjectives as basing form. But it is not intuitive if the most common collocation is in a different gender. The second problem is that many words are ambiguous, there are several different lemmata, in some cases even in different part of speech.

We have solved this problem by modification of the evaluation script. The original procedure computes the compactness of all possible clusters. In our modification there is a preparation phase for each cluster where we find the best matching lemma (or lemma + part of speech, depending on the thesauri) for each word in the query.

The last problem creates multiword units in the queries. These are handled for thesauri using word vectors as the sum of individual words of the multiword. This technique cannot be used for a Sketch Engine thesaurus. The new feature for computing thesaurus for multiword units is in development, we have skipped the queries containing multiwords in the evaluation of for this paper.

4 The New Data Set

To use outlier detection on Czech corpora we have prepared a new data set of outlier detection lists of Czech words. During the creation of the data, two annotators compiled 48 clusters of words in four different languages. Then they translated each cluster to other languages. The result contains each cluster in five languages: Czech, Slovak, English, German, French. We have found many errors (typos, bad translations) in the data, this paper deals with the Czech part only.

An example of the clusters from the new data set is listed in table 1. It shows two clusters in both Czech and English variant. The first cluster contains words which are not in basing form as adjectives (they use feminine gender) or they

are ambiguous (nouns or adjectives). In the second cluster (Electronics) one can see an example of multiword unit (*mp3 player*).

Table 1: Example of two data set clusters – Colors and Electronics

Colors		Electronics	
Czech	English	Czech	English
červená	red	televize	television
modrá	blue	reproduktor	speaker
zelená	green	notebook	laptop
žlutá	yellow	tablet	tablet
fialová	purple	mp3 přehrávač	mp3 player
růžová	pink	mobil	phone
oranžová	orange	rádio	radio
hnědá	brown	playstation	playstation
dřevěná	wooden	blok	notebook
skleněná	glass	sešit	workbook
temná	dark	kniha	book
zářivá	bright	CD	CD
pruhovaný	striped	energie	energy
puntíkovaný	dotted	světlo	light
smutná	sad	papír	paper
nízká	low	ráno	morning

5 Evaluation

We made the evaluation on three Czech corpora: Czes2 (460 million tokens), czTenTen12 (5 billion tokens) [10], csTenTen17 (12 billion tokens) [11].

We have selected only clusters without multiword units, 9 clusters, these form 72 queries. The results are summarized in Table 2. The czTenTen12 corpus was evaluated with Sketch Engine thesaurus and also with word vectors compiled by FastText. We have also included prebuild model from Common Crawl.

Table 2: Evaluation of Czech thesauri on 72 queries of the new data set

	OOP	Accuracy
Czes2	92.2	70.8
czTenTen12	93.4	79.2
csTenTen17	94.3	81.9
czTenTen12 (fasttext)	97.7	87.5
cc.cs	98.1	95.8

6 Conclusions

The outlier detection is probably the best task for evaluating distributional thesauri. This paper describes the new data set of Czech outlier detection lists. We have used this data set on several Czech corpora and we think that this evaluation is suitable for developing new methods and/or optimizing parameters of computing distributional thesauri. We will add more languages to the next version of the new data set.

The comparison of Sketch Engine thesaurus and words vectors generated by FastText shows that FastText provides better results.

Acknowledgements This work has been partly supported by the Czech Science Foundation under the project GA18-23891S.

References

1. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
2. Camacho-Collados, J., Navigli, R.: Find the word that does not belong: A framework for an intrinsic evaluation of word vector representations. In: Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP. pp. 43–50 (2016)
3. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: The concept revisited. In: Proceedings of the 10th international conference on World Wide Web. pp. 406–414. ACM (2001)
4. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. Proceedings of Euralex pp. 105–116 (2004), `http://www.sketchengine.co.uk`
5. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
6. Panchenko, A., Morozova, O., Fairon, C., et al.: A semantic similarity measure based on lexico-syntactic patterns. In: Proceedings of KONVENS 2012 (2012)
7. Rubenstein, H., Goodenough, J.B.: Contextual correlates of synonymy. Communications of the ACM **8**(10), 627–633 (1965)
8. Rychlỳ, P.: Evaluation of the sketch engine thesaurus on analogy queries. In: RASLAN. pp. 147–152 (2016)
9. Rychlý, P., Kilgarriff, A.: An efficient algorithm for building a distributional thesaurus (and other sketch engine developments). In: Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions. pp. 41–44. Association for Computational Linguistics (2007)
10. Suchomel, V.: Recent czech web corpora. In: RASLAN. pp. 77–83 (2012)
11. Suchomel, V.: cstenten17, a recent czech web corpus. In: RASLAN. pp. 111–123 (2018)

A Distributional Multi-word Thesaurus in Sketch Engine

Miloš Jakubíček and Pavel Rychlý

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
{jak,pary}@fi.muni.cz

Lexical Computing
Brno, Czech Republic
{milos.jakubicek,pavel.rychly}@sketchengine.eu

Abstract. In this paper we present an extension of the current distributional thesaurus as available in the Sketch Engine corpus management system towards multi-word units. We explain how multi-word sketches are used to generate multi-word unit candidates, thus preserving access to the underlying corpus texts. Finally we present sample results on the British National Corpus and discuss future development as well as difficulties in evaluation.

Keywords: text corpus, Sketch Engine, MWE, multi-word expressions, thesaurus

1 Introduction

This paper elaborates on a new development implement in Sketch Engine, a leading corpus management system [3], focusing on making a distributional thesaurus for multi-word units available. Since 2006 Sketch Engine features thesaurus for single-word units, calculated using the information obtained from Sketch Engine's word sketches [6]. In 2012, a extension to the word sketch concept has been introduced towards handling multi-word sketches [4]. Since then, the single-word thesaurus basically waited to catch up the multi-word development, so here it is, finally.

In this paper we first describe Sketch Engine, introduce the concept of word sketches and the relation of word sketches to the computation of the distributional thesaurus. We continue by explaining how multi-word sketches are calculated and how they are used to derive a multi-word sketch thesaurus. We conclude by showing preliminary results in the British National Corpus.

2 Sketch Engine

Sketch Engine is a leading text corpus management system which as of 2019 includes several hundreds of preloaded corpora, monolingual as well as parallel

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 143–147, 2019.

ones, available to its users, who can also create their own corpora, have them annotated (part-of-speech tagged, lemmatized etc.) and contrast them against the preloaded ones. The preloaded corpora typically come from the web and are very large. In 2010, Sketch Engine started the so-called TenTen series of web corpora [1], aiming at building a corpus of ten billion words (10^{10}, thus "TenTen") for as many languages as possible.

Targeting ten billion words was not a random choice: by 2010 we had a corpus of that size for English and it clearly showed that it allows many of the Sketch Engine features that work well with a one billion word corpus and single-word units, to work well also on multi-word units. Also, given the Zipfian distribution observed in natural language, it was clear that making the corpora bigger is the only possible way that would allow us to research further on the issues of multi-word expressions.

3 Word sketches

A word sketch is a short summary of a word's collocational behaviour from the perspective of individual grammatical relations (noun's modifier, verb's subject etc.), as can be seen from the example given in Figure 1.

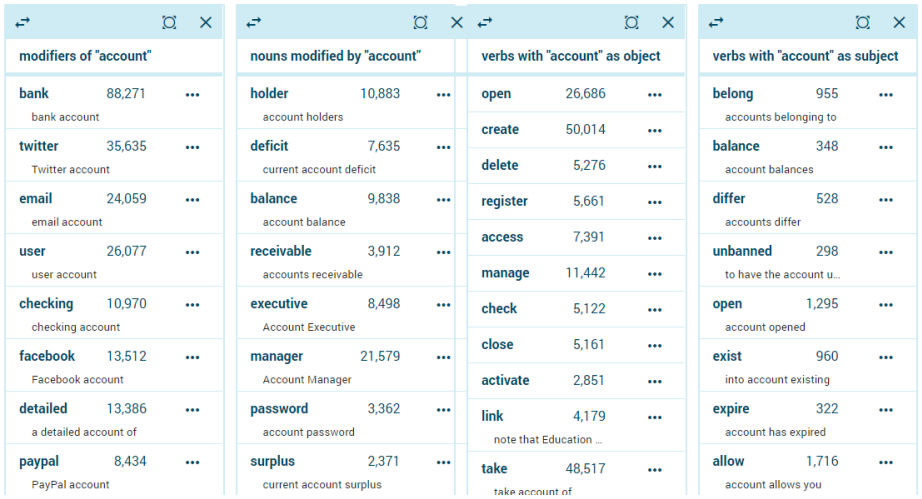


Fig. 1: An example of a word sketch for the English noun *account*.

Each word sketch item is a triple consisting of the headword, the grammatical relation and the collocate. As such a word sketch is basically a dependency syntax graph, calculated using a hybrid rule-based and statistical approach. The backbone word for computing word sketches represents a hand-written word

sketch grammar, which selects collocation candidates using the corpus query language (CQL, [2]).

A sketch grammar typically makes heavy use of regular expressions over morphological annotation of the corpus to select syntactically viable collocation candidates. These candidates are subsequently subject to statistical scoring using a word association score. LogDice is used as the association metric in Sketch Engine as it was proven to be scalable across corpora of different sizes and produces scores comparable across corpora too [5].

In [4], an extension to the word sketch formalism has been presented which allowed the users to obtain a word sketch for multi-word units. That development took a very flexible approach towards multi-word expressions: any two or more words interlinked with word sketch relation formed a multi-word expression for which word sketches were calculated. This method has one strong advantage, namely that it accounted well for discontinuous multi-word expressions (because word sketch relations may catch rather long distance dependencies) and one obvious disadvantage that the words hat to be linked in the sketch grammar, therefore less developed sketch grammars did not provide so many multi-word expressions.

test *(noun)* Alternative PoS: verb (freq: 941,372)
enTenTen [2012] freq = 1,915,482 (147.70 per million)

Lemma	Score	Freq
testing	0.520	558,727
assessment	0.410	640,347
analysis	0.399	1,196,660
procedure	0.382	1,311,372
study	0.380	3,090,402
method	0.373	2,760,051
application	0.366	3,171,582
program	0.365	6,442,955
datum	0.362	3,165,540
evaluation	0.360	468,130
model	0.357	2,557,538
training	0.354	2,486,409
research	0.354	3,171,715
examination	0.352	375,991
requirement	0.349	1,734,482
exam	0.349	373,769
review	0.348	1,803,362

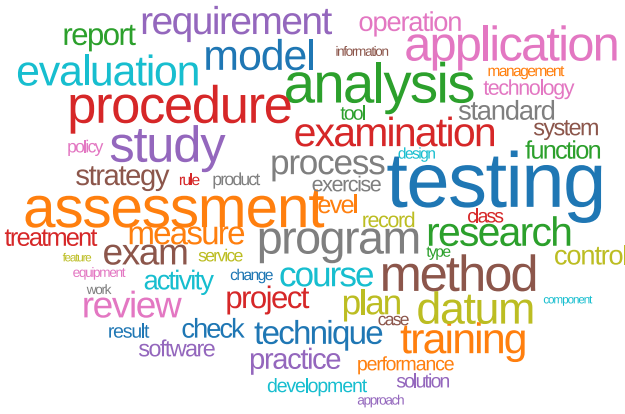


Fig. 2: An example of the thesaurus for the English noun *test*.

4 Thesaurus

On top of the word sketches a distributional thesaurus has been part of Sketch Engine since 2006, facilitating an efficient algorithm which was tracktable

on multi-billion word corpora [6]. The thesaurus is using word sketches for computing the similarity score: it basically compares word sketch collocations for every pair of words in the corpus and the similarity relates to the fraction of shared collocates between these two words, taking the collocation weights as given by logDice into account.[1] A thesaurus screenshot can be found in Figure 2.

The new multi-word extension of the thesaurus uses the multi-word sketches as its backbone. The calculation starts by dumping the whole word sketch database and discovering multi-word sketches (i.e. two and more words connected with a word sketch relation) with a minimum frequency of 100 (less frequent items are unlikely to have any salient thesaurus items). These items form a new multi-word thesaurus lexicon in addition to single-word items, and are subject to the normal thesaurus calculation. In Figure 1 we show a sample multi-word entry obtained from the (by current measures, rather small) British National Corpus [7].

Table 1: Thesaurus items for the phrase "kohl helmut chancellor" on the BNC.

score	frequency	item
1.00	755	kohl chancellor helmut
0.88	1790	kohl helmut
0.75	7307	kohl chancellor
0.34	606	kohl
0.20	140	mitterrand president
0.18	536	chancellor kohl
0.17	340	bush president us
0.17	153	bush us president
0.17	20	re-unification
0.17	116	bush us
0.16	370	bush george president
0.16	283	bush president george
0.16	116	clinton president

5 Conclusions and Future Work

In this paper we have presented an extension allowing the Sketch Engine's thesaurus to be applied to multi-words. The biggest advantage of the approach is that (by relying on the multi-word sketches), it makes very few assumptions

[1] See `https://www.sketchengine.eu/documentation/statistics-used-in-sketch-engine/` for an exact formula.

on the form of the multi-word expressions: any two or more words are allowed as long as they are connected through a sketch grammar relation.

The results immediately indicate further space for improvement: as of now the order of the words in the multi-word expression matters, thus "kohl helmut chancellor" is different (but most similar) to "kohl chancellor helmut", which is not very user-friendly. For production use the phrases will be handled disregarding the word order. The most challenging aspect of the thesaurus development is its evaluation though, whether it concerns single-words or multi-words. Assessing word's similarity is a very difficult task for humans and therefore it is very difficult to obtain reliable evaluation datasets feature high inter-annotator agreement. Nevertheless this is a topic that we want to focus on in the future.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin infrastructure LM2015071 and OP VVV project CZ.02.1.01/0.0/0.0/16_013/0001781. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731015.

References

1. Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., Suchomel, V.: The TenTen Corpus Family. International Conference on Corpus Linguistics, Lancaster (2013)
2. Jakubíček, M., Rychlý, P., Kilgarriff, A., McCarthy, D.: Fast syntactic searching in very large corpora for many languages. In: PACLIC 24 Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation. pp. 741–747. Tokyo (2010)
3. Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., Suchomel, V.: The Sketch Engine: ten years on. Lexicography **1** (2014). https://doi.org/http://dx.doi.org/10.1007/s40607-014-0009-, `http://dx.doi.org/10.1007/s40607-014-0009-9`
4. Kilgarriff, A., Rychlý, P., Kovár, V., Baisa, V.: Finding multiwords of more than two words. Proceedings of EURALEX2012 (2012)
5. Rychlý, P.: A lexicographer-friendly association score. Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN pp. 6–9 (2008)
6. Rychlỳ, P., Kilgarriff, A.: An efficient algorithm for building a distributional thesaurus (and other sketch engine developments). In: Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions. pp. 41–44. Association for Computational Linguistics (2007)
7. Aston, G., Burnard, L.: The BNC handbook: Exploring the British National Corpus with SARA. Edinburgh University Press (1998)

Subject Index

Author Index

RASLAN 2019

Thirteenth 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 2019

ISBN 978-80-263-1530-8
ISSN 2336-4289