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

Ota Mikušek

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

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

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

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

1 Introduction

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

All source code used for the implementation and evaluation is open source[1].

2 Data exploratory analysis

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

[1] https://gitlab.fi.muni.cz/xmikusek/czech-parlament-prediction

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2022, pp. 15–23, 2022.

3 Data preprocessing

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

3.1 Sentences

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

3.2 Lemmatization

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

3.3 Tagging

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

3.4 Syntactic analysis

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

3.5 Additional information

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

4 Classification results

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

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

On two tasks:

1. party classification
2. member classification

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

5 Models

5.1 Baseline (Simple bag of words)

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

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

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

5.2 Bag of words lemmatized

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

Table 1: Party classification results on validation tests

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

Table 2: Party classification on validation tests with tuned parameters

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

Table 3: Member classification results on validation tests

Model folder name	precision (%)	Recall (%)	Classes with precision and recall 0 %
tokens_100000_embeddings_knn	31	29	0
tokens_100000_embeddings_lSVM	23	16	0
tokens_100000_embeddings_naive_bayes	12	15	38
tokens_100000_embeddings_random_forest	30	26	0
tokens_100000_embeddings_SVM	33	26	0
tokens_100000_naive_bayes_all_params	44	22	38
tokens_100000_random_forest	43	20	28
tokens_100000_random_forest_all_params	43	21	30
tokens_100000_random_forest_all_params_balanced	35	23	0
tokens_100000_random_forest_balanced	32	20	0
tokens_100000_simple_bag_of_words	26	27	0
tokens_100000_svc	42	33	0
tokens_100000_svc_all_params	DNF	DNF	DNF
tokens_100000_svc_all_params_balanced	DNF	DNF	DNF
tokens_100000_svc_balanced	36	30	0

Table 4: Member classification on validation tests with tuned parameters

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

Table 5: Classification on test set

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

5.3 Naive Bayes

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

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

5.4 Random forest

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

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

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

5.5 SVM (Support vector machine)

Since SVM was running too long, for party classification (Table 1), regulation parametr C=0.001 was used to make it run faster at the cost less successful classification.[2]

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

5.6 Bag of SVM

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

[2] See `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVM.html` for details.

5.7 Naive Bayes with word embeddings

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

5.8 Random forest with word embeddings

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

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

5.9 SVM with word embeddings

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

5.10 KNN with word embeddings

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

6 Conclusions

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

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

7 Future work

7.1 Data expansion

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

7.2 Comparison with transformers or neural networks

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

7.3 Alternative text segmentation

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

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

References

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

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