# Compressed FastText Models for Czech Tagger

Zuzana Nevěřilová

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

**Abstract.** We are building a new tagger for the Czech language that uses two models: the FastText model for word embeddings and a neural network that assigns tags to tokens. In the deployment, we are struggling with model sizes. Since the model size is a common obstacle in various tasks, several compression methods exist. Authors of the methods often claim that the impact on model performance is minimal. However, the evaluation is done on the two tasks the word embeddings are evaluated on: word analogy and word similarity. No information is provided for the evaluation of subsequent tasks.
In this paper, we have trained a FastText word embedding model on more recent data. We retrained the tagger with the same parameters using compressed and uncompressed variants of the original FastText model and the new one. After comparing the results, we can see quantization methods are suitable, possibly together with pruning, without significant impact on the tagger performance. The precision dropped by 0.1 percentage point only in quantized models. All tested compression methods reduce the model size 10–100 times.

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

## 1 Introduction

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

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

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2022, pp. 79–87, 2022. 

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

## 2 Neural Tagger for Czech

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

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

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

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

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

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

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

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

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

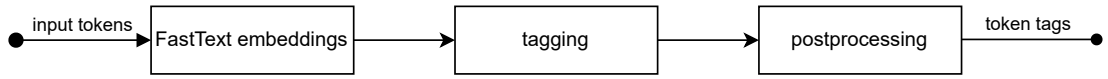


Fig. 1: Tagger processing pipeline

## 3 Different Embedding Models for the Tagger

In the following experiments, we preserve the tagger architecture and all parameters. The only thing that changes is the embedding model. First, we experimented with the pre-trained model for Czech[1]. Second, we experimented with different compression methods. Third, we trained our FastText model for the Czech language from two different language resources: Wikipedia and the SYN corpus. The advantage of the resulting model is that it uses more recent data. On the other hand, the model size is bigger than the original pre-trained model size, even though the training data were smaller. We experimented with the compression of this new model as well. With each FastText model, we retrained the neural tagger for Czech and evaluated it on 10k sentences not present in the training data. The goal is not to have the best parameters for the tagger but to see the impact of different FastText models on tagger performance.

### 3.1 Pre-trained FastText model for Czech

In [5], the authors present pre-trained FastText models for 157 languages. The model for Czech was created from two sources: Wikipedia[2] and CommonCrawl[3]. The former is a collection of high-quality texts, however, the corpus size is relatively small: Czech Wikipedia corpus contained 179 million tokens in 2017, and 785k appeared at least five times. The latter contained 13 billion tokens in 2017, and the vocabulary size of the model was 8.7 million.

### 3.2 New FastText model for Czech

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

[1] Available at `https://fasttext.cc/docs/en/crawl-vectors.html`
[2] `https://www.wikipedia.org/`
[3] `https://commoncrawl.org/`

peared at least five times. For the processing of the Wikipedia data, we used the Wikicorpora package[4].

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

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

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

## 4 Embedding Compression Methods

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

These include:

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

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

[4] `https://github.com/effa/wikicorpora`

[5] `https://fasttext.cc/docs/en/options.html`

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

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

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

### 4.1 Parameters of Uncompressed and Compressed Models

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

Table 1: Model descriptions and sizes

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

[6] `https://github.com/avidale/compress-fasttext`

## 5 Evaluation methods

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

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

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

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

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

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

## 6 Results

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

$$ACT = COR + INC + SPU \qquad P = \frac{COR}{ACT} = \frac{TP}{TP+FP}$$
$$POS = COR + INC + MIS \qquad R = \frac{COR}{POS} = \frac{TP}{TP+FN}$$

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

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

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

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

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

## 7 Conclusion and Future Work

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

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

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

## References

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

3. Chinchor, N., Sundheim, B.: MUC-5 evaluation metrics. In: Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993 (1993), https://aclanthology.org/M93-1007
4. Dale, D.: Compressing unsupervised fasttext models (December 2021), https://towardsdatascience.com/eb212e9919ca, [Online; posted 12-December-2021]
5. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki, Japan (May 2018), https://aclanthology.org/L18-1550
6. Jakubíček, M., Kovář, V., Šmerk, P.: Czech morphological tagset revisited. In: Horák, R. (ed.) Proceedings of Recent Advances in Slavonic Natural Language Processing 2011. pp. 29–42. Tribun EU, Brno (2011), https://nlp.fi.muni.cz/raslan/2011/paper05.pdf
7. Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T.: Fasttext.zip: Compressing text classification models (2016), http://arxiv.org/abs/1612.03651, cite arxiv:1612.03651Comment: Submitted to ICLR 2017
8. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(1), 117–128 (2011). https://doi.org/10.1109/TPAMI.2010.57
9. Křen, M., Cvrček, V., Henyš, J., Hnátková, M., Jelínek, T., Kocek, J., Kováříková, D., Křivan, J., Milička, J., Petkevič, V., Procházka, P., Skoumalová, H., Šindlerová, J., Škrabal, M.: SYN v9: large corpus of written czech (2021), http://hdl.handle.net/11234/1-4635, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University
10. Lioutas, V., Rashid, A., Kumar, K., Haidar, M.A., Rezagholizadeh, M.: Distilled embedding: non-linear embedding factorization using knowledge distillation. CoRR **abs/1910.06720** (2019), http://arxiv.org/abs/1910.06720
11. Nevěřilová, Z., Stará, M.: Neural tagger for czech language: Capturing linguistic phenomena in web corpora. In: Horák, A., Rychlý, P., Rambousek, A. (eds.) The 13th Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2019, Karlova Studanka, Czech Republic, December 6-8, 2019. pp. 23–32. Tribun EU (2019), http://nlp.fi.muni.cz/raslan/2019/paper10-neverilova.pdf
12. Suchomel, V.: cstenten17, a recent czech web corpus. In: Aleš Horák, P.R., Rambousek, A. (eds.) Proceedings of the Twelfth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2018. pp. 111–123. Tribun EU, Brno (2018), https://nlp.fi.muni.cz/raslan/2018/paper10-Suchomel.pdf
13. Šmerk, P., Rychlý, P.: Majka – rychlý morfologický analyzátor. Tech. rep., Masarykova univerzita (2009), http://nlp.fi.muni.cz/ma/

# Appendix

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

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

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

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