QUALITY OF WORD VECTORS AND ITS IMPACT ON NAMED ENTITY RECOGNITION IN CZECH

František Dařena[1], Martin Süss[1]

[1] *Mendel University in Brno, Czech Republic*

EUROPEAN JOURNAL OF BUSINESS SCIENCE AND TECHNOLOGY

Volume 6 Issue 2
ISSN 2694-7161
www.ejobsat.com

ABSTRACT

Named Entity Recognition (NER) focuses on finding named entities in text and classifying them into one of the entity types. Modern state-of-the-art NER approaches avoid using handcrafted features and rely on feature-inferring neural network systems based on word embeddings. The paper analyzes the impact of different aspects related to word embeddings on the process and results of the named entity recognition task in Czech, which has not been investigated so far. Various aspects of word vectors preparation were experimentally examined to draw useful conclusions. The suitable settings in different steps were determined, including the used corpus, number of word vectors dimensions, used text preprocessing techniques, context window size, number of training epochs, and word vectors inferring algorithms and their specific parameters. The paper demonstrates that focusing on the process of word vectors preparation can bring a significant improvement for NER in Czech even without using additional language independent and dependent resources.

KEY WORDS

Named Entity Recognition, word embeddings, word vectors training, natural language processing, Czech language

JEL CODES

C63, C88

1 INTRODUCTION

Named Entity Recognition (NER) is one of the important subtasks of Information Extraction. It focuses on finding named entities in text and classifying them into one of the entity types. The types typically include persons, locations, organizations, temporal expressions, phone numbers, but sometimes also product names, brands, diagnoses, drug types, or pub-

DAŘENA, František, and SÜSS, Martin. 2020. Quality of Word Vectors and its Impact on Named Entity Recognition in Czech. *European Journal of Business Science and Technology,* 6 (2): 154–169. ISSN 2694-7161, DOI 10.11118/ejobsat.2020.010.

lishers (Goyal et al., 2018; Nadeau and Sekine, 2007).

Named entities can be extracted using several approaches. The *knowledge-based*, also known as rule-based approach relies on the availability of various lexicons and domain-specific knowledge (Yadav and Bethard, 2018). Knowledge-based systems can be usually easily implemented but it is difficult to define all necessary rules. The systems usually have high precision but, on the other hand, lower recall and fail on unknown cases.

Machine learning approaches strive to eliminate the problems with hand-crafted rules. The NER problem is being solved with a model automatically created by a computer. Systems using supervised learning require annotated corpora (a text with marked entities) and a learning algorithm that can automatically extract the rules for detecting entities. Systems based on unsupervised learning (no labeled data is available) require only some syntactic patterns to identify candidates for entities that can be further evaluated and disambiguated (Etzioni et al., 2005; Nadeau et al., 2006).

The crucial aspect of learning a NER model is the selection of the appropriate features. Modern state-of-the-art NER approaches avoid using hand-crafted features and rely on feature-inferring neural network systems based on word embeddings. These systems often outperform the systems using engineered features, even when they have access to domain-specific rules or lexicons (Yadav and Bethard, 2018).

A lot of research concentrates on massively used languages, like English, German, or Spanish and there have been many approaches to named entity recognition developed. For the Czech language, the situation is quite different as there is a delay in current research. There exist only a few named entity recognizers and not much attention has been devoted to the optimization of all steps of the NER procedure. A typical example is a process of preparing word vectors to be used in the NER task. The goal of the paper is thus to analyze the impact of different aspects related to word embeddings on the process and results of the named entity recognition task in Czech. The goal is not to achieve the best results and beat the current state-of-the-art approaches, which usually requires using other language-dependent resources, but to discover how different algorithms, their parameters, or the size and quality of corpora used for training can influence the result.

2 CURRENT STATE

The methods of NER often employ statistics (e.g., Conditional Random Fields – see Tkachenko and Simanovsky, 2012; Hidden Markov Models – see Zhou and Su, 2002), classification algorithms (e.g., support vector machine – see Li et al., 2005), or neural approaches (Collobert et al., 2011). In the past, classical machine learning models like SVM or logistic regression strongly relying on feature engineering were popular in NER (Goldberg, 2016). The features generally belonging to one of the three categories – document, corpus, and word-based features (Goyal et al., 2018) usually include, e.g., word length, capitalization, presence in an external list, part-of-speech, position in a sentence, the occurrence of a period or hyphen, suffixes, prefixes, or orthographic features (Zhou and Su, 2002; Tkachenko and Simanovsky, 2012).

Later, it has been found that neural models (especially deep neural models) able to learn important features directly from texts could be used also for NER. A prevalent approach is now based on neural networks with architectures such as bidirectional or convolutional LSTM (Lample et al., 2016; Chiu and Nichols, 2016; Rudra Murthy and Bhattacharyya, 2018; Chen et al., 2018). Such architectures that are suitable for processing sequential data as they have a form of memory are successfully used also in other natural language processing tasks (Mikolov et al., 2015). After a pioneering publication on word vectors training using the word2vec algorithm (Mikolov et al., 2013a), the

NER research was aimed at using word vectors also in NER in many natural languages (Nguyen et al., 2019; El Bazi and Laachfoubi, 2019; Seok et al., 2016). Word vectors (Collobert et al., 2011), which are vectors representing individual words, are able to capture the syntactic as well as semantic regularities of a language which has been found to be beneficial in many NLP tasks.

In order to learn word vectors using a neural model, texts need to be converted to a structured representation (vectors) first. The procedure can generally include several preprocessing steps like, e.g., text cleaning, white space removal, case folding, spelling errors corrections, abbreviations expanding, stemming, stop words removal, or negation handling (Dařena, 2019). For word embeddings training, some preprocessing can be applied too (Li et al., 2017; Leeuwenberg et al., 2016) which can have an impact on the context of the words, the number of unique words, and global word frequencies. Subsequently, one-hot encoded vectors (vector where only one out of its units is 1 and all others are 0) that act as the inputs and outputs of the neural models are derived (Rong, 2014). Various sets of word embeddings trained on different corpora (e.g., Wikipedia) are instantly available. Different algorithms can be also used to train their own set of embeddings, that are suitable for general use or specific task. The algorithms have various parameters that need to be set with respect to a given task. Current approaches to NER using word embeddings, however, often use the default parameters settings, and the impact of alternative settings is not evaluated.

Besides the core features derived from the text in a neural model, additional language-dependent (presence in a list of cities, countries, first and last names, days of a week, currencies, part-of-speech, singular/plural) or language-independent features (context, position, word length, fixed length prefix/suffix, presence of a hyphen) can be on the input of a NER system (Chiu and Nichols, 2016).

2.1 NER for the Czech Language

Ševčíková et al. (2007) presented the first NER system for the Czech language using decision trees analyzing handcrafted features to detect and classify entities in text. Kravalová and Žabokrtský (2009) implemented another system using SVM for classification. Král (2011) implemented a NER system for a specific purpose (searching the Czech press agency database) and demonstrated that feature selection plays a crucial role in designing a NER system. He proved that language independent features are more important than the dependent ones. Konkol and Konopík (2011) created a NER system using the Maximum Entropy algorithm which used semantic spaces that were created using the COALS method (Rohde et al., 2004). It is the first work that treated words as vectors in a multidimensional space. Another system employing Conditional Random Fields using different features and resources was presented by Konkol and Konopík (2013). In the same year, Straková et al. (2013) published another NER system for Czech using Maximum Entropy Markov Model.

The first system that employed word vectors trained using word2vec was presented by Demir and Özgür (2014). Although it used only language independent features it outperformed all existing NER systems for Czech. A better performance was later achieved by Straková et al. (2016) who use a neural network with gated recurrent units together with word vectors representing original or lemmatized words, part-of-speech tags, prefixes, suffixes, or vector representations of characters. The word vectors in both systems were trained using word2vec with the skipgram architecture. The best performance was brought by Konopík and Pražák (2018). They use a deep neural model with LSTM layers encoding character sequences and word sequences together with a wider context information obtained from Latent Dirichlet Allocation. The word sequence layer was using pretrained GloVe and fastText word vectors.

The systems using word vectors were able to improve the performance expressed by the F1-measure by a few percent. At the same time, the features did not need to be engineered manually because many useful properties and relations were encoded in the vectors. Most of the systems, however, relied on pretrained word vectors or created the vectors using default parameters of the algorithms.

2.2 Learning Word Embeddings

In machine learning, there is a general problem with choosing the right set of features for the given task (Blum and Langley, 1997). In natural language processing, there is an additional problem related to the classical representation of features derived from texts (known as bag-of-words). In this model, each word or another feature is represented by one dimension in a multidimensional space for representing the documents. Such a value does not enable sharing some information across features and is thus is independent of the others.

To solve the problem with no similarity among features, it would be possible to add other information to the existing features to better capture the context in which they appear. This, however, increases the number of dimensions in the input space and requires the combination of possible feature components to be carefully selected (Goldberg, 2016).

Some of the modern representations of texts use more dimensions to represent each word or feature. The words are embedded in a continuous multidimensional space that has typically a few hundred dimensions so we talk about word embeddings. Finding suitable values of the vector elements is based on the hypothesis stating that words in similar contexts have similar meanings (Levy and Goldberg, 2014). Because similar words (e.g., synonyms) share some information, the values of their vector elements should be similar and the vectors are located close to each other in the multidimensional space.

Popular approaches leading to generating such vectors include models using global matrix factorization like Latent Semantic Analysis (LSA) or Latent Dirichlet Allocation (LDA) and models learned by neural networks using a small context window (where word2vec is probably the most popular), see Mikolov et al. (2013a), Pennington et al. (2014). Supervised methods create embeddings that are trained towards the given goal and can capture information that is relevant for the task. They, however, require annotated data for the specific task. Unsupervised methods do not require annotated data. Their only goal is to compute embeddings that are usually learned in the task of predicting a word given its context or deciding, whether a word can belong to a context given examples of real and randomly created word-context pairs (Goldberg, 2016). Such embeddings capture general syntactic and semantic relationships and can be applied in a wide variety of tasks. When there is not enough data for domain-specific embeddings training available a model created on a general corpus can be adjusted using a smaller amount of domain-specific data (Yen et al., 2017).

Famous methods that can be used to compute word embeddings include:

- **word2vec** – a family of methods proposed by Mikolov et al. (2013a) that strongly attracted the NLP community to neural language models. The method predicts a word based on its context (the Continuous bag-of-words or CBOW approach) or the context for a word (the skipgram approach). Word2vec tried to eliminate the problems with the computational complexity of the existing neural language models. In the training phase, a neural network uses a linear activation function instead of the sigmoid function, which is typical for a multilayer perceptron, and the logarithm of the probability of predicting the word or its context is being maximized.
- **GloVe** – a model uses information about global co-occurrences of words. Word vectors are used in the task of predicting the probability with which two words co-occur. The probabilities can be calculated from a term-term matrix created from a corpus. The prediction is made by a function that takes word vectors as the input. The word

vectors are calculated in the process of word co-occurrence matrix factorization using stochastic gradient descent (Pennington et al., 2014).

- **fastText** – a model derived from word2vec, treats each word as a bag of character n-grams (which enables considering sub-word information very important for morphologically rich languages) where the vectors are associated at the n-gram level. The vector for a word is calculated as the sum of n-gram vectors. This enables, compared to word2vec and GloVe, creating vectors for words that are not in the training data (Bojanowski et al., 2017).

The skipgram technique in both word2vec and fastText algorithms can better capture semantic regularities of words. On the other hand, the CBOW approach captures syntactic regularities better (Mikolov et al., 2013b).

The methods of embeddings training require several parameters to be set – the number of vector dimensions, definition of the context (size and position), maximal number of unique words, minimal frequency of a word, number of training epochs, etc. which can significantly influence the quality of the learned vectors (Levy et al., 2015).

3 DATA AND METHODS

The quality of word vectors depends on the corpus on which they are trained. Generally, the more data is available, the better. However, the number of unique tokens, amount of errors, writing style, domain to which the texts are related etc. play a significant role also in the NER task. Here, what are an entity and its type often depends on the domain (Kulkarni et al., 2016). The number of unique tokens is especially high for morphologically rich languages where different forms of a word have an impact on the number of global occurrences as well as the number of combinations with other words. Here, normalization techniques, like stemming, lemmatization, case folding, or stop words removal can be considered (Levy et al., 2015).

3.1 Data

For the Czech language, no corpora suitable for the NER task existed before 2007 when the Czech Named Entity Corpus (CNEC) 1.0 was released (Ševčíková et al., 2007). The corpus was later extended, simplified, and transformed to a format similar to the one used by the SIGNLL Conference on Computational Natural Language Learning (CoNLL) and evolved to so-called Extended-CNEC corpus (Konkol and Konopík, 2013). The corpus in version 2.0 defining seven most commonly used entity types (numbers in addresses, geographical names, institution names, media names, artifact names, personal names, and time expressions) was used for training our NER system. This corpus has also been used by most of the other researchers so a comparison with previous research is possible.

To evaluate the impact of corpus size and quality, which are the factors influencing the quality of word vectors (Levy et al., 2015) used in the NER system, different corpora were used to learn word vectors. CWC-11 is a Czech corpus based on selected newspaper articles, blogs, and discussions on the Czech web (Spoustová and Spousta, 2012). CoNLL-2017 is a corpus released for the CoNLL conference in 2017. It contains also documents in Czech, especially from Wikipedia and other Internet sources (Zeman et al., 2018). CZES is a Czech corpus containing data from news webs from the years 1995–1998 and 2002 (Masaryk University, 2011). SYN-2015 is a part of the SYN corpus consisting of journalistic, technical, and fiction papers from the years 2010–2014 (Křen et al., 2016). EuroParl is a relatively small and specialized corpus containing texts related to the European parliament agenda in the years 1996–2011. Detailed characteristics of the data collections can be found in Tab. 1.

Tab. 1: Selected corpora containing open Czech text

Corpus	Number of words	Number of unique tokens
CoNLL-2017 (Czech part)	1.62 billion	21.5 million
CWC-2011 – articles	628 million	1.8 million
CZES	497 million	3.5 million
EuroParl (Czech part)	13 million	304 thousand
Czech Wikipedia extraction	134 million	2.6 million
SYN-2015	121 million	1.4 million
Extended CNEC 2.0	199 thousand	35 thousand

The CoNLL-2017 corpus is the largest but probably of the lowest quality, according to the number of unique tokens. The CWC-2011, CZES, and SYN-2015 corpora contain a lower number of unique tokens so the tokens should appear with higher frequencies. The difference between these corpora is mainly in their size. The EuroParl corpus contains data from a specific domain and is relatively small. Because the CNEC and EuroParl corpora are rather small, the neural network implementing NER is allowed to update the word vectors (the vectors are trainable). Although this is usually not useful, updating word vectors with respect to a specific task might be a good option when the word vectors are not good enough (Hope et al., 2017). This fine-tuning for each task can also give an extra boost to the NER system performance (Collobert et al., 2011). When the other corpora are used to train word vectors, the vectors are fixed during the NER system training.

The Extended CNEC 2.0 corpus, which is primarily used to train the NER system, was used as one of the corpora for learning word vectors. The goal was to find out whether it is beneficial to compute word vectors from the corpus that is also used to train the system for the NER task when there is only a small corpus for training a NER system (with labeled named entities) available as it is expected that different NLP tasks employ the linguistic information related with other tasks (Güngör et al., 2018).

Texts from all sources were lowercased. The reason is that some of the available texts were already in lower case so we wanted to have all of them in the same form. All non-alphanumeric characters and words with less than 5 occurrences were removed as well.

3.2 The NER System

The NER system implemented to evaluate the impact of different properties of word vectors and parameters of their learning were based on the work of Žukov-Gregorič et al. (2018) who achieved state-of-the-art results on the CoNLL NER dataset. The input to the system is a sequence of word vectors and the output is an entity type label (including a label for words that are not entities). The function mapping inputs (a sequence of word vectors) to outputs (a sequence of entity type labels) is a neural network.

The first layer of the network accepts word vectors and passes them to the hidden layer. The hidden layer uses bidirectional LSTM units. The output layer converts the signal from the hidden layer using hierarchical softmax to predict an entity type for the given input. As a stochastic gradient-based optimization algorithm, Adam (Kingma and Ba, 2014) was used to learn the weights of the network. Various hyperparameters of the network were determined experimentally, see below.

Initially, the NER system used pre-trained word vectors as input. The vectors were learned on the Czech part of the CoNLL-2017 collection with word2vec using the skipgram architecture, with context window of size 10, and word vectors having 100 dimensions (Fares et al., 2017).

The values of the hyperparameters (Feurer and Hutter, 2019) of the NER system can significantly influence the results. Because the best possible achievement in the NER task was not the main goal of the research, only an acceptably good setting was found. Initially,

the hyperparameters were set to the values typical for existing research (Žukov-Gregorič et al., 2018). The values were then changed (in both directions) as long as the performance (measured by the F1-measure) of the NER system was improving.

The suitable hyperparameter values were found in the order as they appear in the list below. The suitable number of epochs was found as the average number of epochs that were needed to achieve the best result for the given combination of hyperparameters (this was 12 most of the time).

The best results were achieved with the following hyperparameter setting:

- dropout probability (the probability that a neuron will be randomly turned off): 0.7;
- learning rate (influencing how fast are neuron weights changing): 0.02;
- batch size (the number of instances used for network parameters adjustment in an iteration): 32;
- the number of hidden layer neurons: 200;
- the number of epochs (the number of passes through the training data set): 12.

With this setting, the system was able to achieve the value 0.6816 of the F1-measure on the CONLL test set without optimizing the process word vectors creation.

3.3 Changing Parameters During Word Vectors Training

There are a few aspects of word vectors training. They are evaluated in isolation in a sequence of experiments. In one phase, one parameter is investigated and its suitable value determined. The following phase works with this value and focuses on another parameter.

The corpora described in Section 3.1 were used to learn word vectors to evaluate the impact of different corpora sizes size and quality. The word2vec algorithm using the skipgram architecture, context windows of size 10, hierarchical softmax, minimal token frequency, and the number of epochs equal to 5 was used to learn vectors with 100 dimensions. The skipgram architecture is suitable for most of the NLP tasks, is often used by other authors, and has low computational complexity (Levy et al., 2015).

Another important parameter is the size of vectors. Generally, the bigger the vectors are, the more relations between words can be captured (Pennington et al., 2014). This was, however, demonstrated on the word analogy task and not on the NER task. The vector size is also related to the corpus size. The bigger the corpus is, the more words and relations can be contained there. The experiments, therefore, examined different corpus and vector sizes. Most of the previous works used word vectors with 100 to 300 values. We, therefore, examined 50, 100, 200, 300, and 400 dimensions which cover the mentioned interval as well as close values outside it.

Some of the commonly used text preprocessing techniques, namely lemmatization, case folding, stop words removal, and their combinations were applied to texts before learning word vectors (lemmatization and case folding should be then applied to training data for the NER system too). Lemmatization and case folding belonging to normalization techniques decrease the number of unique tokens and increase the global frequencies of the tokens. This might be important especially for small corpora containing texts in a morphologically rich language, like Czech.

Three algorithms, namely word2vec (using the CBOW and skipgram architectures), GloVe, and fastText (using the CBOW and skipgram architectures) were studied. In the experiments, different context window sizes (5, 10, and 15 words) at a fixed number of epochs (5) were examined. Subsequently, 1, 10, and 15 epochs (5 epochs were already included in the experiments with different context window sizes) of training using 10 words context window were applied to create word vectors.

For the best settings of word2vec and fastText algorithms, the output layer function was changed from softmax to negative sampling with 5 or 10 negative samples. In the word2vec CBOW method, summation was used together with averaging the vectors. Different n-gram sizes were studied for the fastText algorithm. Similarly to Bojanowski et al. (2017), the

minimal n-gram size was 2 or 3 and the maximal size 4 or 6. In the GloVe algorithm, different exponent values in the weighting function were used.

The following list summarizes the investigated aspects of individual techniques and algorithms during word vectors learning:

- all algorithms:
  - corpus: different corpora from Tab. 1;
  - number of dimensions: 50, 100, 200, 300, 400;
  - context window size: 5, 10, 15;
  - preprocessing techniques: lemmatization, case folding, stop words removal;
  - number of training epochs: 1, 5, 10, 15;
- wor2vec and fastText:
  - architecture: skipgram or CBOW;
  - last layer function: hierarchical softmax or negative sampling (5 or 10 negative samples);
- wor2vec:
  - CBOW aggregation: sum or average;
- fastText:
  - character n-gram size: 2 to 6;
- GloVe:
  - value of exponent α in the weighting function in the cost function: 0.75, 0.5, 0.25.

The quality of word vectors can be measured in several ways. One of the popular approaches is the analogy task (Mikolov et al., 2013b). However, good results in this task do not have to automatically lead to good results in the NER task. The performance of NER systems is usually measured using precision and recall. The precision is calculated as the ratio of pieces of a text that were correctly labeled as an entity and the number of pieces of a text that were labeled as an entity. The recall is defined as the ratio of entities in the text that were labeled as entities and the total number of entities in the text. These measures are calculated for each category of entities to be identified and can be further combined to the F1-measure which is a harmonic mean of the precision and recall (Yadav and Bethard, 2018). The impact of changing different parameters during the investigation was thus measured by the F1-measure.

4 RESULTS

This section provides the results from the investigation of the impact of different aspects of word vectors learning. The aspects follow the procedure described in Section 3.3.

4.1 Corpus Characteristics

First, the suitability of different corpora for creating word vectors was evaluated. The improvement against the baseline when no word vectors were used (the input contained just word identifiers) can be found in Tab. 2.

The CoNLL-2017, CZES, and CWC-2011 corpora has brought the highest improvements of the F1-measure (more than 15%) in the NER task even without focusing on the optimization of the parameters of the algorithms used. Among these three, CoNLL-2017 has brought the least improvement despite having the nighest number of tokens. This means that not only the quantity, but also quality of the corpus is important.

Tab. 2: The values of the F1-measure obtained by the NER system when using different corpora for word embeddings training

Corpus	F1-measure [%]
No (baseline)	49.83
EuroParl (Czech part)	52.09
Text from Extended CNEC 2.0	52.54
Czech Wikipedia extraction	56.54
SYN-2015	61.31
CoNLL-2017 (Czech part)	64.60
CZES	65.28
CWC-2011 – articles	66.31

None from the corpora used in word vectors training was able to improve the NER outcomes so they would outperform the result achieved when training the NER system with vectors

pretrained on the Czech part of the CoNLL-2017 corpus, see Section 3.2 for details (the achieved F1-measure was 0.68). This means that it makes sense to focus on the details of the algorithms of word vectors training.

The best results were achieved with the CWC-2011 corpus (a large corpus with more than 600 million words) which is used in the following experiments.

4.2 Corpus Size and Word Vectors Length

The next experiment focused on determining how corpus size and word vectors length are related and how they influence the results of NER. The outcome of this experiment is summarized in Tab. 3.

Tab. 3: The values of the F1-measure of the NER system when using different numbers of word vectors dimensions and sizes of the CWC-2011 corpus (the baseline is emphasized)

Number of dimensions	Relative corpus size			
	1%	10%	50%	100%
50	53.84	59.14	63.98	62.55
100	54.56	61.14	64.83	**66.31**
200	55.17	62.79	66.35	67.29
300	56.63	63.78	67.93	67.95
400	55.70	65.51	67.68	68.33

Most significant differences can be seen between low- and high-dimensional vectors learned on the largest corpus where smaller vectors were not sufficient for encoding all relations between words. Increasing the number of dimensions lead to improvements even for smaller corpus portions. When the dimensionality was around 300 or 400 the results stopped improving, or they even degraded. Improvements were also positively related to corpus size. While the improvements between using 1 and 50% of the corpus were around 10% in the F1-measure, the differences between using 50% of the corpus and the whole corpus were marginal. Finding a suitable amount of data from which the results stop improving had thus a positive effect on computational complexity. For Czech, corpora containing hundreds of million tokens seem to be sufficient.

In the subsequent experiments, the number of dimensions was 300 because it enabled achieving the best results. Further experiments that did not change corpus size worked with 50% of texts from CWC-2011. Both decisions did not negatively influence the performance of the NER system and had favorable computational complexity and memory demands.

4.3 Text Preprocessing Techniques

The application of normalization techniques and stop words removal lead to a decreased number of unique tokens which not only affected word vectors training but also the number of out of vocabulary words (words that are recognized in the testing phase but are unknown in the training phase) in the NER system training. The effects of the application of these techniques and their combinations can be found in Tab. 4. Based on the results, it can be noted that using normalization (here, the largest effect has lemmatization) had a positive impact especially for smaller corpora used for word vectors training. For larger text collections, especially with texts of higher quality, these techniques or their combinations were not useful. Of course, the same preprocessing techniques were applied to the texts used for the NER system training. Lowercasing is considered as a baseline since all texts were lowercased for the initial experiments (an explanation is in Section 3.1).

4.4 Algorithms and Their Parameters

Until now, only the word2vec algorithm was used to train word vectors. In the following step, other algorithms and their parameters were examined. Two parameters were relevant for all three algorithms (word2vec, GloVe, and fastText): context window size and the number of training epochs. The detailed results obtained for different parameter values can be found in Tab. 5. The table contains the values of the F1-measure for different context window sizes for a fixed number of training epochs, as well as the values of the F1-measure for different

Tab. 4: The values of the F1-measure achieved by the NER system and the number of out of vocabulary (OOV) words from 51,092 when using different portions from the CWC-2011 corpus and preprocessing techniques when creating word vectors (LC = lower case, LM = lemmatization, SW = stop words removal)

	F1-measure [%]			Number of OOV words		
Text preprocessing	**Relative corpus size**			**Relative corpus size**		
technique	**50%**	**10%**	**1%**	**50%**	**10%**	**1%**
No	68.70	65.94	56.96	4923	8982	21214
LC (baseline)	67.93	63.78	56.63	4173	7652	18665
LM	67.33	66.35	61.58	2744	4576	10699
SW	67.22	62.56	53.81	5367	9426	21645
LC + LM	66.81	66.22	60.80	2792	4628	10624
LC + SW	65.43	62.61	53.96	4617	8096	19107
LM + SW	68.27	64.38	58.25	2811	4662	10827
LC + LM + SW	66.85	63.23	56.39	2839	4701	10744

numbers of training epochs for a fixed context window size.

It can be seen that fastText dominated in all these experiments. Also the skipgram technique for both word2vec and fastText has brought better results than CBOW, which means that semantic similarity is more important than syntactic one (Mikolov et al., 2013b).

Tab. 5: The values of the F1-measure when using different algorithms and their parameters (context window size, number of training epochs) for word vectors training (the baseline is emphasized)

	Context window size (training epochs = 5)		
Algorithm	**5**	**10**	**15**
word2vec (CBOW)	64.58	62.69	60.53
word2vec (skipgram)	69.06	**68.70**	67.99
GloVe	63.90	64.46	64.54
fastText (CBOW)	66.01	63.91	64.39
fastText (skipgram)	72.47	71.50	72.44

	Training epochs (window size = 10)			
Algorithm	**1**	**5**	**10**	**15**
word2vec (CBOW)	61.46	62.69	60.65	61.18
word2vec (skipgram)	68.25	68.70	68.60	69.05
GloVe	60.43	64.46	63.04	65.00
fastText (CBOW)	64.33	63.91	63.62	63.62
fastText (skipgram)	71.72	71.50	69.36	70.48

A context window size had a larger impact on the results when the CBOW technique was used and only a negligible impact when using the skipgram technique. The number of training epochs seems to have no significant impact on the results. Five epochs of training lead to the best results in most of the experiments. The GloVe algorithm did not reach results comparable to word2vec or fastText even when changing the exponent α in the weighting function in the cost function from the default value 0.75 to 0.5 or 0.25.

Both word2vec and fastText using skipgram can use different functions of the output layer of the neural network – hierarchical softmax or negative sampling. When using 5 or 10 negative samples, no improvement was observed for fastText. On the other hand, five negative samples increased the value of the F1-measure from 69.06% to 70.37% for word2vec. When using the CBOW technique, the sum instead of the average of the output vectors improved the value of the F1-measure from 64.58% to 66.62% for word2vec. For fastText, the value of the F1-measure even decreased. In both cases, the results were worse than when using the skipgram technique. When changing the minimal and maximal n-gram sizes for fastText (minimal = 2 or 3, maximal = 4 or 6), no improvements were found compared to the default setting (minimal = 3, maximal = 6).

4.5 Best Algorithms Settings

After a series of experiments focusing on different aspects of word vectors preparation for the NER task were conducted, some recommendation for the settings have been identified:

Tab. 6: The values of the F1-measure for the NER system using recommended settings for different CWC-2011 corpus portions (the baseline is emphasized)

	No lemmatization				With lemmatization	
	Relative corpus size				Relative corpus size	
Algorithm	**100%**	**50%**	**10%**	**1%**	**10%**	**1%**
word2vec	70.66	**70.37**	66.02	53.78	66.89	59.11
GloVe	63.54	**65.00**	60.12	52.62	58.54	53.44
fastText	71.69	**72.47**	71.12	66.51	70.34	66.42

Tab. 7: Detailed NER performance measures for the best word vectors preparation settings

Entity type	**Precision [%]**	**Recall [%]**	**F1-measure [%]**	**Recognized entities**	**Number of entities**
Numbers and addresses	90.38	85.45	87.85	52	55
Geographical names	73.48	76.98	75.19	396	378
Institutions	59.83	65.74	62.65	356	324
Media	66.67	58.33	62.22	42	48
Artifacts	47.78	47.91	47.84	383	382
Persons	77.49	86.04	81.54	533	480
Temporal expressions	89.63	91.58	90.59	376	368
Total	**70.72**	**74.30**	**72.47**	**2138**	**2035**

- using the CWC-2011 corpus for training (of course, for a specific domain, other corpora might be more suitable; however, the size and quality need to be generally considered);
- training word vectors with 300 dimensions;
- using lemmatization for a small amount of text for training (i.e., 1 or 10% of the corpus);
- word2vec settings: skipgram, the context window size = 5, the number of training epochs = 5, negative sampling with 5 negative samples;
- fastText settings: skipgram, the context window size = 5, the number of training epochs = 5, hierarchical softmax as the output layer function, minimal n-gram size = 3, maximal n-gram size = 6).

The results achieved with these recommendations are summarized in Tab. 6. We can see that lemmatization makes sense in the case that just a small corpus is available for word vectors training. When more data is used, lemmatization even worsens the performance (see the columns 10% in Tab. 6.

The size of the corpus used for training had the smallest impact on the performance when using the fastText algorithm. This supports (Bojanowski et al., 2017) stating that fastText is able to learn well on smaller corpora. From a certain size, using additional texts also did not bring additional improvements in the NER task. From using 50% of the corpus, word2vec was able to provide results comparable to fastText while GloVe was about 5% behind.

Using fastText with other recommended settings of the process and fastText algorithm is therefore the best approach for preparing word vectors for the NER task for the Czech language. The NER system using word vectors prepared in this way achieved 72.47% of the F1-measure (compared to 49.83% when not using word vectors and 68.16% when using setting typically used by other authors).

The detailed performance for individual named entity categories can be found in Tab. 7. The best performance was achieved for temporal expressions (usually the names of days or months) that had very similar vector representations. On the other hand, artifact names, like units of measure, currencies, norms, or product names, were recognized with the worst performance as they cover a very wide variety of expressions. These names were often composed of more tokens and the NER system did not have to recognize all of them correctly.

Compared to word2vec, fastText was able to better recognize numbers and addresses (the F1-measure for fastText was 87.85% compared 70.91% for word2vec). It is complicated to have a separate word vector for every unique token for word2vec. On the other hand, fastText composes word vectors from character n-grams so, for example, a vector for a number will be very similar to vectors of other numbers. A similar situation is in recognizing media entities, that contain many e-mail addresses.

5 DISCUSSION

To evaluate the impact of different settings in word vectors training, the results were compared to the results of other authors that used a similar approach. This means that they used a neural model for NER together with word vectors. The results also needed to be demonstrated on the Extended CNEC 2.0 corpus while using no additional resources (e.g., gazetteers, Brown mutual information bigram clusters, regular expressions) or engineered features (e.g., lemmas, prefixes, affixes, character n-grams, orthographic features). The availability of gazetteers or well-engineered features generally improves the NER system performance so the results of the other authors, against which the comparison is made, are not the best they ever achieved. Not having exactly the same resources also would not allow a direct comparison of the results.

Demir and Özgür (2014) use a large Czech corpus containing 636 million words and 906 thousand unique tokens (the size is similar to the CWC-2011 corpus), together with word2vec using the skipgram architecture and the context window size equal to 5 for training word vectors having 200 dimensions. Straková et al. (2016) used the same algorithm applied to the large SYN corpus for word vectors training. Tab. 8 summarizes the performance of the NER system presented in this paper and the results achieved by other authors where the results were available. The first number 49.83% represents the value of the F1-measure achieved by our system when no word vectors were provided. When word vectors trained using the baseline method (see Section 4.1) were used and suitable values of the NER model hyperparameters were found the value of the F1-measure increased to 68.16%. When focusing on the optimization of different aspects of word vectors training, the value additionally increased to 72.47%.

Tab. 8: Comparison of the values of the F1-measure achieved in this paper with other research

Method	**F1-measure [%]**
This paper – no word vectors	49.83
This paper – word vectors, best hyperparameters	68.16
This paper – best setting for fastText	72.47
Demir and Özgür (2014)	64.72
Straková et al. (2016)	63.91

It is obvious that focusing on the process of word vectors preparation can bring a significant improvement to the NER system performance. This is demonstrated by the comparison to the results of other researchers that did not focus on the optimization of word vectors training. Our results are compared to the outcomes achieved without additional language independent and dependent features and other modifications of the NER algorithm.

6 CONCLUSION

The research focused on named entity recognition in Czech where the process of preparing the data for training a NER model using modern text representations has not been investigated. The main emphasis is put on the phase of preparing word vectors for training a machine learning-based NER system.

First, the NER system inspired by the state-of-art approach was created. The input to the system was a sequence of word vectors and the output was an entity type label. The function mapping inputs (a sequence of word vectors) to outputs (a sequence of entity type labels) was a neural network. The hidden layer used bidirectional LSTM units. The output layer converted the signal from the hidden layer using hierarchical softmax to predict an entity type. As a stochastic gradient-based optimization algorithm Adam was used to learn the weights of the network.

Subsequently, attention was paid to various aspects of word vectors preparation. The suitable settings in different steps were determined in extensive experiments and included the used corpus, number of word vectors dimensions, used text preprocessing techniques, context window size and number of training epochs for word vectors training and other algorithm-specific parameters. Besides suitable values for the parameters, it has been found that a sufficiently large corpus of good quality needs to be used and the number of word vectors dimensions needs to be chosen so enough relations between words can be encoded.

It was demonstrated that focusing on the process of word vectors preparation can bring a significant improvement of the NER system performance even without using additional language independent and dependent resources.

7 REFERENCES

Blum, A. L. and Langley, P. 1997. Selection of Relevant Features and Examples in Machine Learning. *Artificial Intelligence*, 97 (1–2), 245–271. DOI: 10.1016/S0004-3702(97)00063-5.

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5, 135–146. DOI: 10.1162/tacl_a_00051.

Chen, G., Liu, T., Zhang, D., Yu, B. and Wang, B. 2018. Complex Named Entity Recognition via Deep Multi-Task Learning from Scratch. In Zhang, M., Ng, V., Zhao, D., Li, S. and Zan, H. (eds.). *Natural Language Processing and Chinese Computing: Proceedings, Part I*, pp. 221–233. Springer International Publishing, Cham. DOI: 10.1007/978-3-319-99495-6_19.

Chiu, J. P. C. and Nichols, E. 2016. Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4, 357–370. DOI: 10.1162/tacl_a_00104.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12, 2493–2537.

Dařena, F. 2019. VecText: Converting Documents to Vectors. *IAENG International Journal of Computer Science*, 46 (2), 170–177.

Demir, H. and Özgür, A. 2014. Improving Named Entity Recognition for Morphologically Rich Languages Using Word Embeddings. In *2014 13th International Conference on Machine Learning and Applications*, pp. 117–122. DOI: 10.1109/ICMLA.2014.24.

El Bazi, I. and Laachfoubi, N. 2019. Arabic Named Entity Recognition Using Deep Learning Approach. *International Journal of Electrical and Computer Engineering*, 9 (3), 2025–2032. DOI: 10.11591/ijece.v9i3.pp2025-2032.

Etzioni, O., Cafarella, M., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S. and Yates, A. 2005. Unsupervised Named-Entity Extraction from the Web: An Experimental Study. *Artificial Intelligence*, 165 (1), 91–134. DOI: 10.1016/j.artint.2005.03.001.

Fares, M., Kutuzov, A., Oepen, S. and Velldal, E. 2017. Word Vectors, Reuse, and Replicability: Towards a Community Repository of Large-Text Resources. In Tiedemann, J. (ed.). *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pp. 271–276.

Feurer, M. and Hutter, F. 2019. Hyperparameter Optimization. In Hutter, F., Kotthoff, L. and Vanschoren, J. (eds.). *Automated Machine Learning: Methods, Systems, Challenges*, pp. 3–33. Springer International Publishing, Cham.

Goldberg, Y. 2016. A Primer on Neural Network Models for Natural Language Processing. *Journal of Artificial Intelligence Research*, 57, 345–420.

Goyal, A., Gupta, V. and Kumar, M. 2018. Recent Named Entity Recognition and Classification Techniques: A Systematic Review. *Computer Science Review*, 29, 21–43. DOI: 10.1016/j.cosrev.2018.06.001.

Güngör, O., Üsküdarli, S. and Güngör, T. 2018. Improving Named Entity Recognition by Jointly Learning to Disambiguate Morphological Tags. In *Proceedings of the 27th International Conference on Computational Linguistics, Association for Computational Linguistics*, pp. 2082–2092.

Hope, T., Resheff, Y. S. and Lieder, I. 2017. *Learning TensorFlow: A Guide to Building Deep Learning Systems*. O'Reilly Media.

Kingma, D. P. and Ba, J. L. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference for Learning Representations*. CoRR: abs/1412.6980.

Konkol, M. and Konopík, M. 2011. Maximum Entropy Named Entity Recognition for Czech Language. In Habernal, I. and Matoušek, V. (eds.). *Text, Speech and Dialogue: Proceedings*, pp. 203–210.

Konkol, M. and Konopík, M. 2013. CRF-Based Czech Named Entity Recognizer and Consolidation of Czech NER Research. In Habernal, I. and Matoušek, V. (eds.). *Text, Speech, and Dialogue: Proceedings*, pp. 153–160.

Konopík, M. and Pražák, O. 2018. LDA in Character-LSTM-CRF Named Entity Recognition. In Sojka, P., Horák, A., Kopeček, I. and Pala, K. (eds.). *Text, Speech, and Dialogue: Proceedings*, pp. 58–66.

Král, P. 2011. Features for Named Entity Recognition in Czech Language. In Filipe, J. and Dietz, J. (eds.). *Proceedings of the International Conference on Knowledge Engineering and Ontology Development*, Vol. 1, pp. 437–441. DOI: 10.5220/0003660104370441.

Kravalová, J. and Žabokrtský, Z. 2009. Czech Named Entity Corpus and SVM-Based Recognizer. In Li, H. and Kumaran, A. (eds). *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration*, pp. 194–201.

Křen, M., Cvrček, V., Čapka, T., Čermáková, A., Hnátková, M., Chlumská, L., Kováříková, D., Jelínek, T., Petkevič, V., Procházka, P., Skoumalová, H., Škrabal, M., Truneček, P., Vondřička, P. and Zasina, A. J. 2016. SYN2015: Representative Corpus of Contemporary Written Czech. In Calzolari, N., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J. and Piperidis, S. (eds.). *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, pp. 2522–2528.

Kulkarni, V., Mehdad, Y. and Chevalier, T. 2016. Domain Adaptation for Named Entity Recognition in Online Media with Word Embeddings. *arXiv*. CoRR: abs/1612.00148.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. and Dyer, C. 2016. Neural Architectures for Named Entity Recognition. In Knight, K., Nenkova, A. and Rambow, O. (eds.). *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 260–270. DOI: 10.18653/v1/N16-1030.

Leeuwenberg, A., Vela, M., Dehdari, J. and van Genabith, J. 2016. A Minimally Supervised Approach for Synonym Extraction with Word Embeddings. *The Prague Bulletin of Mathematical Linguistics*, 105, 111–142. DOI: 10.1515/pralin-2016-0006.

Levy, O. and Goldberg, Y. 2014. Neural Word Embedding as Implicit Matrix Factorization. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D. and Weinberger, K. Q. (eds.). *Advances in Neural Information Processing Systems 27*, pp. 2177–2185.

Levy, O., Goldberg, Y. and Dagan, I. 2015. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, 3, 211–225. DOI: 10.1162/tacl_a_00134.

Li, Q., Shah, S., Liu, X. and Nourbakhsh, A. 2017. Data Sets: Word Embeddings Learned from Tweets and General Data. In *Proceedings of the Eleventh International AAAI Conference on Web and Social Media*, pp. 428–436.

Li, Y., Bontcheva, K. and Cunningham, H. 2005. SVM Based Learning System for Information Extraction. In Winkler, J., Niranjan, M. and Lawrence, N. (eds.). *Deterministic and Statistical Methods in Machine Learning*, pp. 319–339. DOI: 10.1007/11559887_19.

Masaryk University, NLP Centre. 2011. *czes*. LINDAT/CLARIN Digital Library at the Institute of Formal and Applied Linguistics [online]. Available at: `http://hdl.handle.net/11858/00-097C-0000-0001-CCCF-C`.

Mikolov, T., Chen, K., Corrado, G. S. and Dean, J. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations*. CoRR: abs/1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. 2013b. Distributed Representations of Words and Phrases and Their Compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. O. (eds.). *Proceedings of the 26th International Conference on Neural Information Processing Systems*, Vol. 2, pp. 3111–3119.

Mikolov, T., Joulin, A., Chopra, S., Mathieu, M. and Ranzato, M. 2015. Learning Longer Memory in Recurrent Neural Networks. In *3rd International Conference on Learning Representations*. CoRR: abs/1412.7753.

Nadeau, D. and Sekine, S. 2007. A Survey of Named Entity Recognition and Classification. *Lingvisticæ Investigationes*, 30 (1), 3–26. DOI: 10.1075/li.30.1.03nad.

Nadeau, D., Turney, P. D. and Matwin, S. 2006. Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity. In Lamontagne, L. and Marchand, M. (eds.). *Advances in Artificial Intelligence: Proceedings*, pp. 266–277. DOI: 10.1007/11766247_23.

Nguyen, A.-D., Nguyen, K.-H. and Ngo, V.-V. 2019. Neural Sequence Labeling for Vietnamese POS Tagging and NER. In *Proceedings: 2019 IEEE-RIVF International Conference on Computing and Communication Technologies*. DOI: 10.1109/RIVF.2019.8713710.

Pennington, J., Socher, R. and Manning, C. D. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

Rohde, D. L. T., Gonnerman, L. M. and Plaut, D. C. 2004. An Improved Method for Deriving Word Meaning from Lexical Co-Occurrence. *Cognitive Psychology*, 7, 573–605.

Rong, X. 2014. word2vec Parameter Learning Explained. *arXiv*. CoRR: abs/1411.2738.

Rudra Murthy, V. and Bhattacharyya, P. 2018. A Deep Learning Solution to Named Entity Recognition. In Gelbukh, A. (ed.). *Computational Linguistics and Intelligent Text Processing: Revised Selected Papers*, Part I, pp. 427–438.

Seok, M., Song, H.-J., Park, C.-Y., Kim, J.-D. and Kim, Y.-S. 2016. Named Entity Recognition using Word Embedding as a Feature. *International Journal of Software Engineering and its Applications*, 10 (2), 93–104. DOI: 10.14257/ijseia.2016.10.2.08.

Spoustová, J. and Spousta, M. 2012. A High-Quality Web Corpus of Czech. In Calzolari, N., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Moreno, A., Odijk, J. and Piperidis, S. (eds.). *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pp. 311–315.

Straková, J., Straka, M. and Hajič, J. 2013. A New State-of-the-Art Czech Named Entity Recognizer. In Habernal, I. and Matoušek, V. (eds.). *Text, Speech, and Dialogue: Proceedings*, pp.68–75.

Straková, J., Straka, M. and Hajič, J. 2016. Neural Networks for Featureless Named Entity Recognition in Czech. In Sojka, P., Horák, A., Kopeček, I. and Pala, K. (eds.). *Text, Speech, and Dialogue: Proceedings*, pp. 173–181.

Ševčíková, M., Žabokrtský, Z. and Krůza, O. 2007. Named Entities in Czech: Annotating Data and Developing NE Tagger. In Matoušek, V. and Mautner, P. (eds.). *Text, Speech and Dialogue: Proceedings*, pp. 188–195.

Tkachenko, M. and Simanovsky, A. 2012. Named Entity Recognition: Exploring Features. In *Proceedings of KONVENS 2012*, Vol. 5, pp. 118–127.

Yadav, V. and Bethard, S. 2018. A Survey on Recent Advances in Named Entity Recognition from Deep Learning Models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2145–2158.

Yen, A.-Z., Huang, H.-H. and Chen, H.-H. 2017. Fusing Domain-Specific Data with General Data for In-Domain Applications. In *Proceedings of the International Conference on Web Intelligence* pp. 566–572. DOI: 10.1145/3106426.3106473.

Zeman, D., Hajič, J., Popel, M., Potthast, M., Straka, M., Ginter, F., Nivre, J. and Petrov, S. 2018. CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 1–21. DOI: 10.18653/v1/K18-2001.

Zhou, G. and Su, J. 2002. Named Entity Recognition using an HMM-Based Chunk Tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 473–480. DOI: 10.3115/1073083.1073163.

Žukov-Gregorič, A., Bachrach, Y. and Coope, S. 2018. Named Entity Recognition with Parallel Recurrent Neural Networks. In Gurevych, I. and Miyao, Y. (eds.). *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Vol. 2: Short Papers, pp. 69–74. DOI: 10.18653/v1/P18-2012.

AUTHOR'S ADDRESS

František Dařena, Department of Informatics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic, e-mail: frantisek.darena@mendelu.cz

Martin Süss, Department of Informatics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic