Towards the evaluation of feature embedding models of the fusional languages

Alina Wróblewska, Katarzyna Krasnowska-Kieraś, and Piotr Rybak

Institute of Computer Science, Polish Academy of Sciences
Jana Kazimierza 5, 01-248 Warsaw, Poland
alina@ipipan.waw.pl, kasia.krasnowska|piotr.cezary.rybak@gmail.com

Abstract

The main component of a neural network approach is a dense vector representation of features, i.e. feature embedding. Various feature types are possible, e.g. words, part-of-speech tags. In this paper we investigate what should be used as features in estimating embedding models of the fusional languages – tokens or lemmata. Furthermore, we research the methodological question whether the results of the intrinsic evaluation of embeddings are informative for downstream applications, or the embedding models should be evaluated extrinsically. The presented experiments are conducted on Polish – a fusional Slavic language with a relatively free word order. However, the evaluation results can be approximately generalised to other Slavic languages, because the studied problems are common to them.

1. Introduction

Neural networks are very successful in various language processing tasks, e.g. dependency parsing (Chen and Manning, 2014; Kiperwasser and Goldberg, 2016; Andor et al., 2016), sentiment analysis (e.g. Iyyer et al., 2015). The main component of a neural network is a dense vector representation of features, i.e. feature embedding. Various feature types are possible, e.g. word forms, part-of-speech tags. The particular feature types are represented as vectors of presumably different number of dimensions (more dimensions for word features and less for part-of-speech features). Features represented as d-dimensional vectors are embedded into a d-dimensional space. Different feature types require different vector spaces.

According to the concept of feature embeddings, some features sharing common information should have similar vectors and thus feature similarities can be captured. For example, some Slavic languages make a clear distinction between perfective and imperfective aspects, and contain pairs of the imperfective and perfective verbs, e.g. in Russian съедать vs. съесть, in Polish ZJADAĆ vs. ZJEŚĆ ('to eat'). As the verbs marked for different aspects tend to share most of their valence and selectional preferences, they should be represented with similar vectors.

From the perspective of foregoing research and provided experimental results, it is indisputable that embedding models of the isolating languages, such as English, can be trained on tokens (e.g. Mikolov et al., 2013a; Pennington et al., 2014). Other languages largely adopt procedures of estimating and evaluating feature embeddings proposed for English. However, it is not obvious what should be used as features in estimating embeddings of the fusional languages[1] – tokens, lemmata, or maybe stems.

In experiments on Russian, the embedding models are estimated on stems (Leviant and Reichart, 2015) or tokens simplified with some language-specific rules (Vulić et al., 2017). The embedding models are then intrinsically evaluated on a dataset[2] which is a translation of English SimLex999 (Hill et al., 2015) and consists of pairs of lemmata re-scored by Russian speakers. In experiments on Polish (Mykowiecka et al., 2017), the embedding models are trained on tokens[3] or lemmata, and tested on pairs of lemmata from the publicly available thesauri or on a lemma-based analogy dataset.

Therefore, the first research question addressed in our paper is what should be used as features in training the embedding models of the fusional languages – lemmata or tokens[4] (see Section 2.)? Lemma embeddings seem to be advantageous from the point of view of the intrinsic evaluation. Token embeddings, in turn, seem to be helpful for downstream applications. The second research problem concerns the validation methodology. Are the results of the intrinsic evaluation of embeddings sufficiently informative for downstream applications (see Section 3.)? Or should the embedding models be evaluated *in vivo* in the realistic scenarios (see Sections 4. and 5.)? The presented evaluation experiments are conducted on Polish – a fusional Slavic language with a relatively free word order. However, the evaluation results can be approximately generalised to other Slavic languages, because the studied problems are common to them.

2. Feature embeddings

2.1. Embedding models

When considering a fusional language such as Polish, it is important to make a distinction between possible types of embeddings, including vector representations of tokens, lemmata, inflections, and character n-grams.

[1] According to Aikhenvald (2007:4), in "*fusional* – sometimes misleadingly called (in)flectional – *languages* there is no clear boundary between morphemes, and thus semantically distinct features are usually merged in a single bound form or in closely united bound forms". For example, the suffixes -ами in Russian домами ('house'.inst.pl) and *-ami* in Polish *domami* ('house'.inst.pl) fuse the case and number information.

[2] http://www.leviants.com/ira.leviant/MultilingualVSMdata.html#SimLex999

[3] Tokens are automatically lemmatised, in order to enable the comparison with the test sets.

[4] Throughout this paper, we use the following terms: *lemma* for a word's base form (e.g. KOT 'cat'); *token* for a string of characters in running text (e.g. *kota* 'cat'); *inflectional form/inflection* for a token assigned a morphosyntactic interpretation (e.g. *kota*.subst:acc:sg:m2, *kota*.subst:gen:sg:m2).

Token embedding model A model for tokens is the most straightforward to obtain: the only required resource and tool are a possibly big collection of texts and a tokeniser. In the resulting embedding space, each distinct token obtains its own vector. For instance, there is no single vector corresponding to the "concept" of POCIĄG 'train', but separate vectors for each distinct token of POCIĄG that appears in the training data: *pociąg* ('train'.sg:nom|acc), *pociągu* ('train'.sg:gen|loc), *pociągami* ('train'.pl:inst) etc. Note that under this approach, all syncretic inflectional forms of the same lemma (as well as those homonymous between distinct lemmata) receive a common embedding.

Lemma embedding model Another type of model, more tool/resource-dependent, is an embedding mapping lemmata into a vector space. In order to induce such a model, either an existing large lemmatised corpus or a lemmatiser is necessary. The text fed to the model while training is then a sequence of lemmata assigned to consecutive tokens. The resulting embedding model shares vectors between text occurrences belonging to the same lemma rather than to syncretic/homonymous word forms, but its quality will depend on the quality of the corpus annotation or the lemmatiser.

Inflection embedding model A third possible approach to embeddings for a fusional language is to employ a morphosyntactically annotated corpus or a tagger in order to induce vectors for inflections, i.e. tokens paired with their respective morphosyntactic interpretations. This approach is the most resource/tool costly and, since automatic morphosyntactic tagging of Polish is a difficult task that still requires improved solutions (Kobyliński and Kieraś, 2016), it seems the most prone to propagation of errors.

Subword embedding model Apart from models trained on word forms, there are some experiments on training embeddings on parts of words, e.g. character n-gram embedding model (Bojanowski et al., 2016). This model is dedicated for morphologically rich languages with large vocabularies and a high rate of rare words. To estimate a subword embedding model, a large collection of tokenised texts is required. Tokens are represented as bags of character n-grams. Character n-grams are first represented as vectors and then the representation of each token is estimated as the sum of the appropriate n-gram embeddings, e.g. for $n = 5$, the token *pociągami* ('train'.inst:pl) is represented as the sum of the vectors of <*poci*, *pocią*, *ociąg*, *ciąga*, *iągam*, *ągami*, *gami*>, and <*pociągami*>.

2.2. Experimental setup

We examine token embeddings $\mathcal{T}$, lemma embeddings $\mathcal{L}$, and subword embeddings $\mathcal{S}$ in our studies. All tested vectors are estimated on the same collection of textual data from Polish Wikipedia and National Corpus of Polish (Przepiórkowski et al., 2012). The details of preprocessing (tokenisation and lemmatisation) the dataset are described in (Mykowiecka et al., 2017). As training data are either tokenised or lemmatised, the vocabulary sizes of feature embedding models estimated on these datasets differ considerably. The vocabulary sizes of token, lemma, and subword embeddings are 2.12M, 1.55M and 5M, respectively.

Token and lemma embeddings We use the pre-trained token $\mathcal{T}_d$ and lemma $\mathcal{L}_d$ embeddings[5] (Mykowiecka et al., 2017), for *d* being the vector size. The vectors were derived with Gensim[6] (Řehůřek and Sojka, 2010) – a Python implementation of CBOW and Skip-gram models (Mikolov et al., 2013a). The vectors selected for our experiments were estimated with the following parameters:

- dictionary parameters: trimming the items occurring less than 5 times in training data,
- training parameters: the vector size of 100 or 300; the context window of 5; 10 training epochs; optimisation heuristics: hierarchical softmax or negative sampling; negative samples of 5.

Subword embeddings The subword vectors $\mathcal{S}_d$ are estimated with *fastText* library[7] (Bojanowski et al., 2016). FastText is an extension of the continuous Skip-gram model (Mikolov et al., 2013a). The subword embeddings are trained with the following parameters:

- dictionary parameters: trimming the items less frequent than 5; character n-grams for $3 \leq n \leq 6$,
- training parameters: the vector size of 100 or 300; the context window of 5; 10 training epochs; negative sampling; negative samples of 5.

3. Evaluation methodology

There are two standard ways of evaluating the embedding models: intrinsic evaluation and extrinsic evaluation.

3.1. Intrinsic evaluation

Intrinsic evaluation consists in testing a model against a dedicated test set. There are two standard benchmarks for word embeddings: determining word similarity/relatedness (e.g. Finkelstein et al., 2002; Hill et al., 2015) and word analogy solving (e.g. Mikolov et al., 2013b). The word similarity evaluation estimates the semantic proximity of two words (e.g. with cosine), and correlates this score with the human judgements of the word similarity. The methodology of similarity testing has recently been criticized (e.g. Faruqui et al., 2016; Chiu et al., 2016), because of the uncertain indication of the impact of embeddings on downstream applications.

The word analogy method, in turn, assumes that linear relations between pairs of words (e.g. *king–man* and *woman–queen*) are indicative of the embedding quality. According to this method, the vector of one word (e.g. *queen*) can be estimated of the vectors of the remaining words (i.e. *king - man + woman*). This methodology is questioned as well (e.g. Linzen, 2016), because "information not detected by linear offset may still be recoverable by a more sophisticated search method, and thus is actually encoded in the embedding" (cf. Drozd et al., 2016).

[5] dsmodels.nlp.ipipan.waw.pl

[6] https://radimrehurek.com/gensim

[7] https://github.com/facebookresearch/fastText

Despite criticism this evaluation method is still widely used. Regarding the intrinsic evaluation of Polish embeddings, an extensive study of synonymy and analogy recognition is presented in (Mykowiecka et al., 2017). According to their results, the intrinsic evaluation is appropriate for investigated tasks. However, there are no clues that it is informative for other NLP tasks which to a lesser extent rely on synonymy and analogy recognition.

3.2. Extrinsic evaluation

Extrinsic evaluation consists in integrating a model into a sophisticated NLP task and verifying the impact on the results of this task. According to our knowledge, there are no experiments on evaluating Polish embeddings extrinsically. Two extrinsic evaluation settings are thus considered here: morphosyntactic disambiguation (Section 4.) and dependency parsing (Section 5.).

4. Morphosyntactic disambiguation

The first extrinsic evaluation context considered in this paper is that of morphosyntactic disambiguation (MD). The task of morphosyntactic disambiguation is formulated as follows: given a sequence of tokens together with their respective sets of possible morphosyntactic tags (most commonly obtained from a morphosyntactic analyser), select a single, correct tag for each token. For tokens unknown to the analyser, the set contains a special ign tag and the task amounts to generating a tag instead of selecting one from a provided set. Table 1 shows an example token sequence corresponding to the sentence *Mieszka w Kotkowicach.* '(S)he lives in Kotkowice.', together with possible tags and correct tags that a disambiguator should choose for each token. The performance of an MD tool is measured in terms of accuracy, i.e. the percentage of tokens assigned the same tag as in gold standard data.

token	possible tags	correct tag
Mieszka	subst:sg:gen:m3 fin:sg:ter:imperf subst:sg:acc:m1 subst:sg:gen:m1	fin:sg:ter:imperf
w	prep:acc:nwok prep:loc:nwok	prep:loc:nwok
Kotkowicach	ign	subst:pl:loc:n
.	interp	interp

Table 1: A token sequence with possible and correct tags.

In order to examine the impact of embedding vectors on MD accuracy, the vectors from $\mathcal{T}$ models are used as input features for a morphosyntactic disambiguator.[8] The basic disambiguator has at its core a two-layer, bi-directional LSTM network taking as input a sequence binary-valued vectors representing the sets of possible morphosyntactic tags of consecutive tokens. For a more detailed description of the disambiguator, see (Krasnowska-Kieraś, 2017). For each cosidered token embedding model, the basic input for each token is extended by concatenation with its embedding vector.

The PolEval task 1(A) training dataset is used (Ogrodniczuk et al., 2017) and following measures are calculated for each feature configuration:

- accuracy in 10-fold cross-validation (91.59% for the basic model),
- Δ: difference wrt. accuracy on folds' training data (a measure of overfitting; 2.6 for the basic model).

The results of the experiment are given in Table 2.

MD	100		300	
	acc	Δ	**acc**	Δ
+$\mathcal{T}$ (skipg-hs)	94.54%	2.6	94.97%	2.7
+$\mathcal{T}$ (skipg-ns)	94.77%	2.5	95.15%	2.7
+$\mathcal{T}$ (cbow-hs)	95.01%	2.2	**95.22%**	2.4
+$\mathcal{T}$ (cbow-ns)	94.90%	**1.9**	95.14%	2.2
+$\mathcal{S}$ (skipg-ns)	94.36%	2.5	94.81%	2.7

Table 2: Accuracy and Δ in 10-fold cross-validation for 100- and 300-dimensional embeddings estimated with Skip-gram (skipg) or CBOW (cbow) and optimised with hierarchical softmax (hs) or negative sampling (ns).

Augmenting the morphological information with vector embeddings brings about a substantial gain in accuracy, resulting in error reduction wrt. the basic model ranging from 35% to 43%. For each tested embedding model and configuration (rows in Table 2), the performance of the disambiguator with $\mathcal{T}_{300}$ vectors is consistently better than with $\mathcal{T}_{100}$ vectors, whereas the shorter vectors yield smaller values of Δ. In the MD context, the Skip-gram models perform better when trained with negative sampling while the CBOW ones work better when hierarchical softmax is employed, the latter yielding the best accuracy of 95.22% with $\mathcal{T}_{300}$ vectors.

5. Dependency parsing

Dependency parsing is another task in which feature embeddings are extrinsically evaluated. The main goal of dependency parsing is to derive a syntactic analysis (represented as a directed tree) of a sequence of words, based on their tokens, lemmata, part-of-speech tags, morphological characteristics, and potentially other features, e.g. feature embeddings. We verify two scenarios: 1) the syntactic analysis of some Polish sentences provided by a dependency parser with the linear classifier (baseline) vs. dependency parsers with the neural network classifiers; 2) analysing the Polish sentences using parsers integrated with pre-trained feature embeddings. The quality of the dependency parsers tested in these two scenarios is measured with two standard metrics: UAS[9] and LAS[10].

[8] The $\mathcal{L}$ models are not considered here since correct lemmata for tokens are typically not yet determined at the MD stage of text processing for Polish.

[9] UAS (*unlabelled attachment score*) is the average accuracy of assigning the correct head for each token in each sentence. We provide both the micro- and marco-average scores.

[10] LAS (*labelled attachment score*) is the average accuracy of assigning both the correct head and label for each token in each sentence.

5.1. Dependency parsing systems

Three dependency parsing systems are tested in our experiments: MaltParser (Nivre et al., 2006), SyntaxNet (Andor et al., 2016), and BIST parser (Kiperwasser and Goldberg, 2016).[11] All of them are transition-based parsers, but they differ in the internal classifiers (linear vs. non-linear), and in the feature extractors (hand-crafted vs. feed-forward neural network vs. biLSTM).

MaltParser MaltParser's classifier is based on the logistic regression model (i.e. linear model) and its hand-crafted feature model is built of the single features (e.g. tokens, lemmata, part-of-speech tags) and double or triple combinations of these single features.

SyntaxNet The parser uses a simple feed-forward neural network with two hidden layers of 1,024 dimensions each to make the transition decisions (i.e. non-linear classifier). A transition is based on a rich set of discrete features (i.e. tokens, part-of-speech tags, dependency arcs and labels in the surrounding context of the state, *k*-best tags) extracted in the current parse configuration. The feature set is fed into the neural network, the first layer of which transforms the sparse, discrete features into a dense, continuous embedded representation. SyntaxNet is an extension of the approach by (Chen and Manning, 2014).

BIST parser The feature extraction in BIST parser is based on a biLSTM encoder that is jointly trained with the parser (end-to-end training). Each token is represented as a biLSTM vector taking into account the sentential context of the token. A concatenation of a few biLSTM encodings is used as the feature vector which is passed to the non-linear scoring function (multi-layer perceptron).

5.2. First evaluation experiment

In the first experiment, the parsing results provided by the baseline parser – MaltParser – are compared with the results of the neural network parsers – SyntaxNet and BIST parser. The results of 5-fold cross-validation can be found in Table 3. BIST Parser achieves the best results and outperforms the other two parsers. Since SyntaxNet performs only slightly better than MaltParser, it is not used in our further experiments.

parser	LAS		UAS	
	macro	**micro**	**macro**	**micro**
MALT	81.56%	77.78%	86.07%	82.56%
SyntaxNet	81.64%	77.91%	87.55%	84.23%
BIST	**83.01%**	**79.70%**	**89.07%**	**85.95%**

Table 3: Parsing results provided by MaltParser with the linear classifier, and SyntaxNet and BIST parser with the neural network classifiers.

5.3. Second evaluation experiment

In this experiment, we test whether integrating the parsers with the pre-trained feature embeddings increases the parsing quality. In order to augment MaltParser with token embeddings, the feature model of MaltParser is extended in a naive way: for the top token on the stack and the first token in the buffer we add the number of additional features corresponding to the dimensionality of the token embedding (each value of the token embedding as a separate feature). The results can be found in Table 4. There is a negligible increase in the parsing quality, but only if 100-dimensional token embeddings are used. We suspect that due to its linearity the logistic regression model cannot fully take advantage of the token embeddings.

MALT	LAS		UAS	
	macro	**micro**	**macro**	**micro**
BASELINE	81.56%	77.78%	86.07%	82.56%
$+\mathcal{T}_{100}$	**81.65%**	**77.97%**	**86.08%**	**82.64%**
$+\mathcal{T}_{300}$	80.75%	76.95%	85.43%	81.88%

Table 4: 5-fold cross-validation of MaltParser integrated with token embeddings estimated with Skip-gram and optimised with negative sampling.

As previously noted BIST parser uses the deep learning model to predict the correct sequence of transitions (i.e. to parse a sentence). Each word in the sentence gets its own embedding which corresponds to the hidden state of biLSTM. The embedding of a word corresponds to the concatenation of the vectors of the token and the part-of-speech tag of this word. By default both embeddings are trained from scratch during the parser training. However, it is also possible to apply external embeddings. We test whether augmenting BIST parser with the pre-trained token, lemma, or character *n*-gram embeddings increases its accuracy. Results of the experiment are in Table 5.

BIST	LAS		UAS	
	macro	**micro**	**macro**	**micro**
$\mathcal{T}_{intern}$	83.01%	79.70%	89.07%	85.95%
$+\mathcal{T}_{100}$	**85.70%**	**82.68%**	**90.29%**	**87.52%**
$+\mathcal{T}_{300}$	85.52%	82.46%	90.09%	87.30%
$\mathcal{L}_{intern}$	82.94%	79.85%	89.07%	86.06%
$+\mathcal{L}_{100}$	84.24%	81.27%	89.64%	86.78%
$+\mathcal{L}_{300}$	84.23%	81.16%	89.54%	86.64%
$+\mathcal{S}_{100}$	85.48%	82.48%	90.07%	87.25%
$+\mathcal{S}_{300}$	85.21%	82.24%	89.72%	86.93%

Table 5: 5-fold cross-validation of BIST parser with internal token embeddings ($\mathcal{T}_{intern}$), 100- or 300-dimensional external token embeddings ($\mathcal{T}_{100}$, $\mathcal{T}_{300}$), internal lemma embeddings ($\mathcal{L}_{intern}$), external lemma embeddings ($\mathcal{L}_{100}$, $\mathcal{L}_{300}$), and external character *n*-gram embeddings ($\mathcal{S}_{100}$, $\mathcal{S}_{300}$). All external vectors are estimated with Skip-gram and optimised with negative sampling.

According to the results, external embeddings increase the accuracy of BIST parser. Similarly as for MaltParser, the best results are achieved with 100-dimensional token embeddings, however the improvement is now much more apparent. Even if the results of BIST parsers with internal token and lemma embeddings are comparable, the results of the parser with external lemma embeddings are consistently lower than the results of the parser with the to-

[11] All parsers are trained on the extended version of Polish Dependency Bank (Wróblewska, 2014), which consists of over 16K dependency trees.

ken embeddings. Finally, BIST parser integrated with the character n-gram embeddings performs slightly worse than the parser enhanced with the token embeddings.

Feature embeddings estimated with two methods (CBOW and Skip-gram) and two optimisation techniques (hierarchical softmax and negative samplings) are tested in this experiment. Due to lack of space, we cannot provide all results. We therefore reported only results of BIST parser integrated with external 100-dimensional token embeddings (see Table 6). The best performing BIST parser is enhanced with token embeddings estimated with Skip-gram method and optimised with negative sampling.

BIST	LAS		UAS	
$+\mathcal{T}_{100}$	**macro**	**micro**	**macro**	**micro**
(cbow-ns)	85.57%	82.49%	90.18%	87.36%
(cbow-hs)	85.45%	82.38%	90.02%	87.21%
(skipg-ns)	**85.70%**	**82.68%**	**90.29%**	**87.52%**
(skipg-hs)	85.25%	82.18%	89.93%	87.14%

Table 6: 5-fold cross-validation of BIST parser with external 100-dimensional token embeddings estimated with CBOW (cbow) or Skip-gram (skipg), and optimised with negative sampling (ns) or hierarchical softmax (hs).

6. Conclusions

Methods of evaluating feature embeddings intrinsically have recently been criticised by the NLP community, e.g. because of the uncertain indication of the impact of embeddings on downstream applications. The intrinsic evaluation is undoubtedly appropriate for synonymy and analogy recognition. However, it does not provide clues for other NLP tasks, such as morphosyntactic disambiguation and dependency parsing, which to a lesser extent rely on synonymy and analogy recognition. For these tasks the extrinsic evaluation seems to be reasonably informative.

Using external token embeddings in morphosyntactic disambiguation task brings about a substantial gain in accuracy (improvement by about 4 pp), resulting in error reduction up to 43%. However, the choice of the most appropriate token embedding is a secondary issue, because embeddings of different lengths estimated with different training methods and optimisation techniques have a comparable impact on the task. Similarly, external feature embeddings (especially token embeddings) play an important role in dependency parsing. Even though BIST parser augmented with 100-dimensional token embeddings estimated with Skip-gram method and optimised with negative sampling outperforms parsers with other external embeddings, the differences are actually negligible. Finally, external lemma embeddings are useless for morphosyntactic disambiguation and less useful for dependency parsing than external token embeddings.

Acknowledgments *The presented research was supported by SONATA 8 grant no 2014/15/D/HS2/03486 from the National Science Centre Poland.*

7. References

Andor, D., C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins, 2016. Globally Normalized Transition-Based Neural Networks. In *Proceedings of ACL 2016*.

Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov, 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*.

Chen, D. and C. Manning, 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of EMNLP 2014*.

Chiu, B., A. Korhonen, and Sampo S. Pyysalo, 2016. Intrinsic Evaluation of Word Vectors Fails to Predict Extrinsic Performance. In *Proceedings of RepEval 2016*.

Drozd, A., A. Gladkova, and S. Matsuoka, 2016. Word Embeddings, Analogies, and Machine Learning: Beyond *King−Man+Woman=Queen*. In *Proceedings of COLING 2016*.

Faruqui, M., Y. Tsvetkov, P. Rastogi, and C. Dyer, 2016. Problems With Evaluation of Word Embeddings Using Word Similarity Tasks. In *Proceedings of RepEval 2016*.

Finkelstein, L., E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, 2002. Placing Search in Context: The Concept Revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Hill, F., R. Reichart, and A. Korhonen, 2015. SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation. *Computational Linguistics*, 41(4):665–695.

Iyyer, M., V. Manjunatha, J. Boyd-Graber, and H. Daumé III, 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of ACL-IJCNLP 2015*.

Kiperwasser, E. and Y. Goldberg, 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Kobyliński, Ł. and W. Kieraś, 2016. Part of speech tagging for Polish: State of the art and future perspectives. In *Proceedings of CICLing 2016*.

Krasnowska-Kieraś, K., 2017. Morphosyntactic disambiguation for Polish with bi-LSTM neural networks. In *Proceedings of LTC'17*.

Leviant, I. and R. Reichart, 2015. Separated by an Uncommon Language: Towards Judgment Language Informed Vector Space Modeling. *CoRR*, abs/1508.00106.

Linzen, T., 2016. Issues in evaluating semantic spaces using word analogies. In *Proceedings of RepEval 2016*.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, 2013a. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of NISP 2013*.

Mikolov, T., W. Yih, and G. Zweig, 2013b. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of NAACL HLT 2013*.

Mykowiecka, A., M. Marciniak, and P. Rychlik, 2017. Testing word embeddings for Polish. *Cognitive Studies*, (17).

Nivre, J., J. Hall, and J. Nilsson, 2006. MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *Proceedings of LREC 2006*.

Ogrodniczuk, M., Ł. Kobyliński, and A. Wawer, 2017. Results of the PolEval 2017 Competition: Part-of-Speech Tagging and Sentiment Analysis Shared Tasks. In *Proceedings of LTC'17*.

Pennington, J., R. Socher, and C. Manning, 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of EMNLP 2014*.

Przepiórkowski, A., M. Bańko, R. L. Górski, and B. Lewandowska-Tomaszczyk (eds.), 2012. *Narodowy Korpus Języka Polskiego*. Warsaw: Wydawnictwo Naukowe PWN.

Řehůřek, R. and P. Sojka, 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the Workshop on New Challenges for NLP Frameworks*.

Vulić, I., N. Mrkšić, R. Reichart, D. Ó Séaghdha, S. Young, and A. Korhonen, 2017. Morph-fitting: Fine-Tuning Word Vector Spaces with Simple Language-Specific Rules. In *Proceedings of ACL 2017*.

Wróblewska, A., 2014. *Polish Dependency Parser Trained on an Automatically Induced Dependency Bank*. Ph.D. dissertation, ICS PAS, Warsaw.