Morphosyntactic disambiguation for Polish with bi-LSTM neural networks

Katarzyna Krasnowska-Kieraś

Institute of Computer Science, Polish Academy of Sciences
Jana Kazimierza 5, 01-248 Warsaw, Poland
kasia.krasnowska@gmail.com

Abstract

The paper presents a system for morphosyntactic disambiguation of Polish texts based on a bi-directional LSTM recurrent neural network. The system competed in the PolEval 2017 evaluation campaign task 1(A), achieving the top accuracy of 94.6% on the task's test data.

1. Introduction and related work

Part of speech (POS) tagging is an important task in NLP, necessary in the initial steps of many text processing pipelines. While POS tagging for English has already been extensively researched, with the accuracy of state-of-the-art systems exceeding 97%,[1] the performance of tools available for Polish still calls for a substantial improvement (for an overview and discussion of the currently available and used tagging/disambiguation tools and resources for Polish, see Kobyliński and Kieraś, 2016).

A main source of difficulty and potential confusion when Polish POS tagging is concerned arises from the inflectional nature of Polish. First, the rich inflectional characteristics of word forms call for an elaborate, positional tagset (with ca. 4000 distinct tag combinations as compared to, e.g., 37 tags in the WSJ tagset or 85 tags in the Brown Corpus tagset), making the task more complicated. Second, POS tagging for Polish is typically coupled with morphosyntactic analysis (restricting the sets of possible tags for each token via dictionary lookup), making it less straightforward to define the exact scope of the tagging problem. The firm distinction between the tasks of morphosyntactic tagging and morphosyntactic disambiguation of inflectional languages and a precise definition of both tasks was postulated in the evaluation paper by (Radziszewski and Acedański, 2012).

The first, now obsolete, taggers for Polish were described by (Dębowski, 2004) and (Piasecki, 2007). The research in this field was continued with different approaches, including Brill (Acedański, 2010), memory-based (Radziszewski and Śniatowski, 2011), and CRF (Waszczuk, 2012; Radziszewski, 2013) taggers.

2. Morphosyntactic disambiguation task

The problem of morphosyntactic disambiguation (MD) as considered in this paper is formulated as follows. The input consists of a sequence of sequences (chunks/sentences/paragraphs) of tokens, each token annotated with its morphological analysis (set of its possible morphosyntactic tags, typically obtained from a dictionary or a morphosyntactic analyser) or a special symbol ign, denoting an unknown word with no information about possible morphosyntactic tags available. The task is to output sequences of correct tags corresponding to the input tokens in each input chunk and amounts to:

- selecting the appropriate tag from the list for the tokens for which it is provided,
- generating the appropriate tag for the ign tokens.

Each tag consists of POS followed by values of grammatical categories pertaining to this particular POS. As an illustration and running example, consider the sentence *Mieszka w Kotkowicach.* '(S)he lives in Kotkowice.' The possible tags for the 4 tokens constituting this sentence, as well as the tags correct in this context, are given in Table 1. The first token *Mieszka* is the most ambiguous and can be interpreted as a form of the verb *mieszkać* 'to live', the noun *mieszek*.M3 'moneybag' or the noun *Mieszko*.M1 (male name). The second token, *w*, is unambigous as far as part of speech is concerned, but can be interpereted as a preposition requiring either accusative or locative case. The third token *Kotkowicach* is a name of a village unknown to the morphological analyser and it is up to the disambiguation tool to guess its correct tag. Disambiguation of the last token, punctuation, is trivial as only one possible tag is proposed.

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: An example 4-token sentence with morphological analysis and gold-standard tags.

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.

3. System description

This section provides details of the proposed MD system. 3.1. describes the architecture of a recurrent neural network at the core of the disambiguator. In 3.2., the format

[1] https://aclweb.org/aclwiki/POS_Tagging_(State_of_the_art)

and information content of input vectors fed to the neural network are explained. Finally, 3.3. provides details on final steps transforming the networks's output into the tag sequence returned by the tool.

3.1. Neural network architecture

Morphosyntactic disambiguation is performed by a bidirectional recurrent LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 1999) neural network. The number of hidden layers in the network and the size of hidden LSTM units is a configurable parameter of the system. The network is implemented in Python, using the Keras[2] library with TensorFlow[3] backend. The network structure is schematically shown in Figure 1.

For each text chunk to be morphosyntactically disambiguated, its tokens are translated into lists of input vectors, each vector dedicated to other type of information about the token. The exact content of the input vector list depends on the feature scheme selected in a particular model configuration. Possible input vectors are described in 3.2.

For each token, its input vectors are passed to appropriate subnetworks. Most of these subnetworks simply copy their input to output, but some perform a non-trivial operation (see, e.g., 3.2.2.). The outputs of the subnetworks are then concatenated into one vector that is passed to the first bi-LSTM layer of the network.

The output of the last bi-LSTM layer serves as common input to a number of dense layers with softmax output activation and categorical cross-entropy loss function. Each dense layer predicts a probability distribution over values of a particular part of the correct tag: there is one layer dedicated to the POS, and one for each morphosyntactic category in the tagset. For the morphosyntactic category layers, apart from values of this category found in the tagset, the additional 'absence' value is possible to represent the lack of this particular category in the tag.

3.2. Input vectors

3.2.1. Morphological vector

The morphological information consists of the set of possible tags for each token. The proposed system treats the value at each position (corresponding to POS and particular grammatical categories; tag positions are colon-separated, as shown in Table 1) of each possible tag as a separate information unit. The set of possible tags is then represented as a 0-1 "bag-of-values" (BOV) vector, with the one bit representing each distinct POS/category value in the tagset and additional bits for each category representing its absence.[4] The presence of given category value in any of the possible tags is reflected by the corresponding vector entry being 1. If no possible tag contains given category, its 'absence' bit is set to 1. The unknown tokens are assumed to have the POS value of ign and no morphosyntactic categories specified, so that their vectors can follow the same convention. For example, the morphology vector for the token *Mieszka* in Table 1 would have ones in positions corresponding to subst, fin, sg, acc, gen, m1, m3, ter and imperf, ones in positions corresponding to absence of degree, negation, and other categories non-applicable to subst or fin POS, and zeroes elsewhere.

Note that this BOV representation is lossy: it ignores interactions between particular values and makes some possible tags combinations undistinguishable.[5] On the other hand, it is compact and allows to represent any number of possible tags as a fixed-length vector. An interesting alternative to be explored in further work would be to induce embeddings either for individual tags (and combine them algebraically or via a recurrent subnetwork) or for whole sets of possible tags (e.g. from a large, morphologically analysed but not disambiguated, corpus).

3.2.2. Word embedding

The first additional vector that can be incorporated into the network input is an externally trained word embedding model $\mathcal{M}$.[6] In the case of tokens for which an embedding vector is present in $\mathcal{M}$, the procedure is straightforward, with that same vector being concatenated with the rest of input. In the case of a token t unknown to $\mathcal{M}$ a following heuristic is employed:

- the vocabulary of $\mathcal{M}$ is searched for the set S_t of words sharing the longest possible (non-empty) suffix with t,
- if S_t is not empty, its subset S'_t of words with the lowest Levenshtein distance to t is selected and the average of vectors for words in S'_t is considered to be an approximation of the vector for t,
- if S_t is empty, a random vector is used.

As an illustration, consider the token *Kotkowicach* from our running example. The word embedding models used in our experiments do not account for this text form, but they do provide vectors for *Motkowicach* and *Gotkowicach*, which both happen to be locative forms of city/village toponyms. As both those words are at the same Levenshtein distance from *Kotkowicach*, their two respective vectors are averaged to provide an approximate embedding of *Kotkowicach*.

	token types		token occurrences	
in model	135918	94.80%	983006	80.87%
heuristic	7276	5.07%	8511	0.70%
random				
– interp	34	0.02%	223510	18.39%
– other	149	0.10%	486	0.04%

Table 2: Breakdown of the suffix heuristic use.

Table 2 summarises the applicability of the described heuristic with the 1.2 million token dataset and Word2Vec models used in the experiments presented in 4. The table provides counts of token types and token occurences

[2] https://keras.io

[3] www.tensorflow.org

[4] Sets of possible values of each POS/category are mutually disjoint.

[5] E.g., x:y:z,u:v:w and x:v:z,u:y:w receive identical BOV representations.

[6] Embeddings trained jointly with the disambiguation model are not considered in this work since the currently available annotated training data of 1.2 million tokens is negligibly small when compared to the amounts of data typically used to obtain word embedding models.

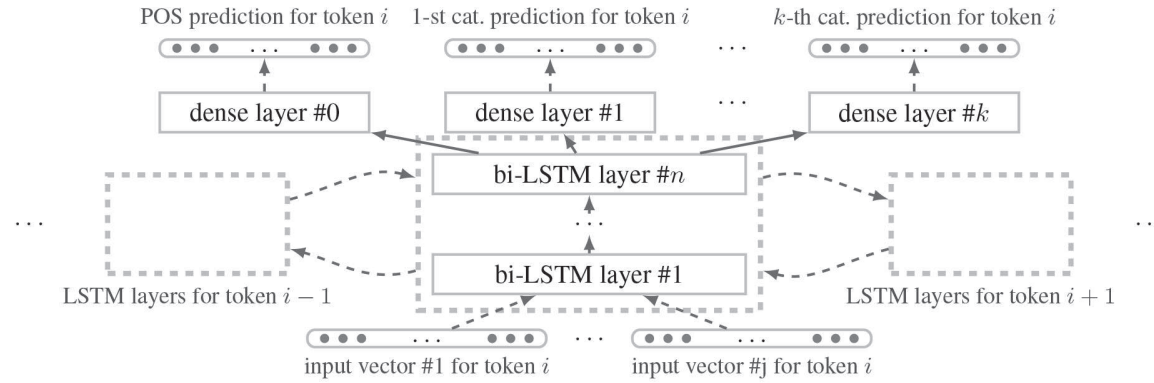

Figure 1: A schema of the network structure.

split into those for which a vector was readily available in the model, for which the heuristic was applicable and for which the fallback to random vector was necessary. Upon closer examination of the last group, it can be observed that a vast majority of its tokens are punctuation marks, trivial to disambiguate with only one possible tag interp. It can therefore be concluded that the potentially non-trivial tokens with random vectors constitute only 0.1% of types and 0.04% of occurrences.

3.2.3. Suffix embedding

As an alternative to embeddings for whole words, we also explore the idea of training vector representations of word suffixes of fixed length (suffix length being a configurable parameter). While the suffix is less informative than the whole token form, it might prove to be sufficient for the purpose of morphosyntactic disambiguation as it roughly corresponds to the inflectional ending of the word. Reducing the word to its last few characters significantly reduces the number of vectors to be learned and increases the number of occurrences of each 'word' in the training corpus. For example, the training data used in this work (see 4.) contains 143473 distinct tokens, with an average of 8.47 occurrences per token. When the tokens are trimmed to the last 4 characters (or the whole token if it has 4 or less characters), those numbers drop to 25931 and 46.87 respectively. Not only are suffix embeddings cheaper to learn in terms of total size of the resulting vectors, but also can be expected to need less training data. Therefore, contrarily to the word embeddings discussed in 3.2.2., the suffix embeddings are trained jointly with the disambiguation model (by an intermediate embedding layer added to the network if the system configuration includes suffix embedding input), and the corresponding input vector only contains one value: the index of the token's suffix on the precomputed suffix list.

3.3. Output correction

The last performed step is to reconstruct actual tags from the network's outputs. Output vectors corresponding to predicted probability distributions over POS and each grammatical category are examined and the values assigned the highest probability are collected (the special 'absence' values are discarded) and combined into a colon-separated string. At this point, it is not yet correct to refer to the resulting string as "morphosyntactic tag" since it may happen to be malformed (inconsistent with the tagset): the network may have made a mistake by selecting an illegal set of categories for the POS. Moreover, the string might be a valid tag, but not one of the possible tags provided for the token.

Both cases are heuristically solved by comparing the generated string with the set of 'candidate' tags (possible tags when available, all valid tags in case of ign segments) and selecting the 'candidate' tag least divergent from the considered string. The divergence measure used in this procedure is Levenshtein distance calculated between encoded versions of tags. The encoding is intended to make the textual similarities/dissimilarities between values found in tags systematic and more meaningful. To this purpose, each POS/category value is mapped to a 2-character sequence, with sequences pertaining to different categories containing different characters, and sequences pertaining to the same category sharing only the first character. For example, nom and acc, both values of case category, might be mapped to `T#` and `T$`, while pos and com, both values of degree category, mapped to `!c` and `!v`.

4. Experiments and evaluation

For experiments described in this paper, the PolEval 2017[7] task 1(A) training dataset ($\approx$ 1.2 million segments, 85660 sentences) was used. The dataset consists of segments annotated with morphosyntactic analysis (i.e. possible tags or ign for each segment) as well as gold-standard tags to evaluate against.

Each model was trained for 20 epochs with batches of 2048 sentences (grouped by number of segments).

Table 3 shows the results obtained in 10-fold cross-validation on the PolEval training dataset. For each tested model configuration, the size of the hidden LSTM layer (two layers were used in all experiments) and the feature scheme is given. Different combinations of following input vectors were tested:

- MORF: morphological vectors (see 3.2.1.),
- W2V$_i$: Word2Vec (Mikolov et al., a; Mikolov et al., b) word embeddings (see 3.2.2.) with vectors of

[7] `http://poleval.pl`

LSTM size	feature scheme	$\mathbf{acc}_G$	$\mathbf{acc}_D$	acc	error red.	$\mathbf{acc}_{train}$	Δ
384	MORF	38.73%	89.23%	**91.59%**	0.0%	94.23%	2.6
384	MORF+W2V$_{50}$	65.65%	93.04%	**94.78%**	38.0%	96.90%	2.1
384	MORF+W2V$_{100}$	67.41%	93.37%	**95.03%**	41.0%	97.18%	2.1
384	MORF+W2V$_{200}$	68.53%	93.62%	**95.22%**	43.1%	97.45%	2.2
512	MORF+W2V$_{200}$	68.22%	93.73%	**95.26%**	43.7%	98.14%	2.9
384	MORF+W2V$_{300}$	68.66%	93.72%	**95.28%**	43.8%	97.60%	2.3
512	MORF+W2V$_{300}$	68.70%	93.75%	**95.30%**	44.1%	98.24%	2.9
384	MORF+S$_3$	62.84%	91.06%	**93.59%**	23.8%	97.25%	3.7
384	MORF+S$_4$	62.97%	91.54%	**93.85%**	26.9%	98.12%	4.3
384	MORF+S$_5$	60.51%	91.65%	**93.81%**	26.5%	98.67%	4.9
384	MORF+R$_{100}$	52.80%	90.95%	**93.11%**	18.1%	95.89%	2.8

Table 3: Results of 10-fold cross-validation on training data. The configuration used in PolEval competition is highlighted.

length i, trained on full National Corpus of Polish (Przepiórkowski et al., 2012) and Wikipedia,[8]
- R$_{100}$: "random" word embeddings of length 100: each word appearing in the data is assigned a vector of random values, drawn uniformly from $[-1; 1]$,
- S$_i$: embeddings for suffixes of length i, with embedding vectors of length 64 (see 3.2.3.).

For each configuration, the following measures were calculated and are presented in Table 3 (weighted average through folds):
- **acc**$_G$: tag-guessing accuracy, i.e. accuracy calculated only for ign segments (4.19% of dataset),
- **acc**$_D$: actual disambiguation accuracy, i.e. accuracy calculated only for genuinely ambiguous segments (those with at least 2 possible tags; 54.28% of dataset),
- **acc**: total accuracy, calculated for all segments,
- **acc**$_{train}$: total accuracy on the fold's training data,
- Δ = **acc**$_{train}$ − **acc**: a measure of overfitting).

The most basic model with single LSTM size of 384,[9] operating solely on morphological information, achieves an overall accuracy of 91.59%, with a very poor guessing accuracy of 38.73%. Adding the word embedding vectors to the feature scheme (while keeping the network size unchanged) brings about a raise in accuracy to 94.78–95.28% depending on the embedding length, with an error reduction of 38.0–43.8% wrt. the basic model and a much improved guessing accuracy in the range of 65.65–68.66%. Moreover, the relative difference between training and testing accuracy Δ decreases to 2.1–2.3 as compared to 2.6 for the basic model. For embedding lenghts of 200 and 300, additional experiments with a larger network (LSTM size of 512) were performed. The respective increases in testing accuracy were very slight while both Δ valued increased to 2.9, which prompts us to favour the less overfitting models.

The improvement seen with suffix embeddings is also visible, although less pronounced: very similar accuracies of 93.50–93.81% and error reduction of 23.8–27.0%. While using embeddings for different suffix lengths does not seem to heavily influence the testing accuracy, the training accuracy raises with suffix length, resulting in gradually higher values of Δ, all exceeding the one calculated for the basic model.

In order to gain some more insight into the impact of word embeddings on MD performance, an experiment with a random model was performed. The random model assigns distinct vectors to distinct words, but does not carry any more information about word similarities, textual distribution etc. It is in this respect similar to the "naïve" one-hot representation, but more compact and therefore more affordable to test. While the random embeddings behave worse than the trained ones (as expected), it is interesting to see that they do improve on the basic model, especially when tag guessing is concerned (with an accuracy increase from 38.73% to 53.80%).

5. Comparison with other results

Given the specific nature of the PolEval 1(A) task, it is difficult to find figures in literature that could be contrasted with our results in a straightforward fashion. We nevertheless provide some previous results reported in two overview papers on Polish morphosyntactic tagging: (Radziszewski and Acedański, 2012) and (Kobyliński and Kieraś, 2016), referred to as EVAL-2012 and EVAL-2016. In both papers, the same annotaded corpus was used as that provided as 1(A) PolEval training data, and all accuracy measures were calculated via 10-fold cross validation. However, the task at hand was different and involved a complete process leading from plain text to tagged tokens (segmentation, morphological analysis and morphosyntactic disambiguation). We cite the following reported measures:
- **acc**$_{dis}$: disambiguation accuracy; very similar to our **acc** measure, although the authors are not clear on whether or how this measure covers guessing performance,
- **acc**$_{upper}$: upper bound tagging accuracy; a tagging[10] accuracy measure assuming all segmentation errors[11] made by the tagger to be correctly tagged,

[8] (Wawer, 2015), `http://mozart.ipipan.waw.pl/~axw/models/orth/`.

[9] A bi-LSTM layer consists of two LSTMs for processing the input left-to-right and right-to-left.

[10] From plain text input.

[11] Tokens with boundaries inconsistent with gold standard.

system/evaluation	EVAL-2012				EVAL-2016	
	$\mathbf{acc}_{dis}$	$\mathbf{acc}_{upper}$	$\mathbf{acc}_{lower}$	$\mathbf{acc}^{U}_{lower}$	$\mathbf{acc}_{lower}$	$\mathbf{acc}^{U}_{lower}$
Pantera (Acedański, 2010)	92.95%	89.09%	88.79%	14.70%	88.95%	15.19%
WMBT (Radziszewski and Śniatowski, 2011)	93.00%	87.82%	87.50%	13.57%	90.33%	60.25%
WCRFT (Radziszewski, 2013)	–	–	–	–	90.76%	53.18%
Concraft (Waszczuk, 2012)	–	–	–	–	91.07%	58.81%

Table 4: Previously reported performance of Polish taggers.

- $\mathbf{acc}_{lower}$: lower bound tagging accuracy; the percentage of tokens that were correctly segmented and tagged;
- $\mathbf{acc}^{U}_{lower}$: $\mathbf{acc}_{lower}$ restricted to ign tokens.[12]

The accuracy measures (where provided) given in the two overview papers are collected in Table 4. All the cited tools, contrarily to our system, do not employ any additional data apart from the morphosyntactically annotated training corpus.

Among works concerning succcessful use of bi-LSTM (and other) neural networks for tagging, (Plank et al., 2016) report bi-LSTM taggers trained and tested on Universal Dependencies data (POS only, 17 tags) for multiple languages, with an accuracy of 97.63% for Polish.

It is also worthwhile to cite results for Czech, which is a closely related (both genetically and typologically) language with a very similar tagset size and two-step approach to morphosyntactic tagging (morphological dictionary lookup followed by morphosyntactic disambiguation). The Czech state-of-the art status is currently held by MorphoDiTa (no additional external data, 95.75% accuracy, Straková et al., 2014) and Morče (semi-supervised training, 95.89% accuracy, Spoustová et al., 2009).

6. Portability to other languages

The described system was only tested on Polish, but its general design allows to easily accommodate it for other tagsets or languages. It was however not our aim to implement a universal tool, but rather one suited to the specifics of MD of Polish, therefore several assumptions were made. First, the system is designed to operate on large, positional tagsets typical for inflectional languages. Second, a morphological analysis step is expected to have been performed earlier in the text processing pipeline. Third, the possibility of including information such as word embeddings or other tool's prediction in the input vectors depends of course on the availability of suitable tools and resources.

7. Conclusions

A new method for morphosyntactic disambiguation of Polish was described in this paper. The proposed system builds on a bi-LSTM recurrent neural network and outperformed all the other systems competing in PolEval task 1(A). The design of the presented morphosyntactic disambiguator should allow for its use for other inflectional languages with a processing pipeline similar to the Polish one.

[12] A more lenient measure (an analogue of $\mathbf{acc}_{upper}$) would be more suitable for comparison, but was not provided.

Acknowledgements *The presented research was supported by SONATA 8 grant no 2014/15/D/HS2/03486 from the National Science Centre Poland. The author would like to thank Łukasz Dębowski (ICS PAS) for sharing his ideas for the network architecture and data representation.*

8. References

Acedański, S., 2010. A morphosyntactic Brill tagger for inflectional languages. In *Advances in Natural Language Processing*.

Dębowski, Ł., 2004. Trigram morphosyntactic tagger for Polish. In *Proceedings of the International IIS: IIPWM'04 Conference*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Gers, F. A., J. Schmidhuber, and F. Cummins, 1999. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12:2451–2471.

Hochreiter, S. and J. Schmidhuber, 1997. Long Short-term Memory. In *Neural computation*, volume 9.

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

Mikolov, T., K. Chen, G. Corrado, and J. Dean, a. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR 2013*.

Mikolov, T., I. Sutskever, K. Chen, G. Corrado, and J. Dean, b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS 2013*.

Piasecki, M., 2007. Polish tagger TaKIPI: Rule based construction and optimisation. *Task Quarterly*, 11(1–2):151–167.

Plank, B., A. Søgaard, and Y. Goldberg, 2016. Multilingual part-of-speech tagging with bidirectional Long Short-Term Memory models and auxiliary loss. In *Proceedings of ACL 2016 (Volume 2: Short Papers)*. Association for Computational Linguistics.

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

Radziszewski, A., 2013. A tiered CRF tagger for Polish. In *Intelligent Tools for Building a Scientific Information Platform: Advanced Architectures and Solutions*. Springer Verlag.

Radziszewski, A. and S. Acedański, 2012. Taggers gonna tag: an argument against evaluating disambiguation capacities of morphosyntactic taggers. In *Proceedings of TSD 2012*, LNCS. Springer-Verlag.

Radziszewski, A. and T. Śniatowski, 2011. A Memory-Based Tagger for Polish. In *Proceedings of LTC 2011*.

Spoustová, D., J. Hajič, J. Raab, and M. Spousta, 2009. Semi-supervised training for the averaged perceptron POS tagger. In *Proceedings of EACL 2009*. Athens, Greece: Association for Computational Linguistics.

Straková, J., M. Straka, and J. Hajič, 2014. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of ACL 2014: System Demonstrations*. Baltimore, Maryland: Association for Computational Linguistics.

Waszczuk, J., 2012. Harnessing the CRF complexity with domain-specific constraints. The case of morphosyntactic tagging of a highly inflected language. In *Proceedings of COLING 2012*. Mumbai, India.

Wawer, A., 2015. Sentiment dictionary refinement using word embeddings. In *Proceedings of ISMIS 2015*. Cham: Springer International Publishing.