Data Sparsity in Highly Inflected Languages: The Case of Morphosyntactic Tagging in Polish

Author: Michael Ustaszewski

Supervisors: Rodrigo Agerri and German Rigau

Master's Thesis

Master in Language Analysis and Processing
University of the Basque Country (Euskal Herriko Unibertsitatea)
Department of Computer Systems and Languages

Donostia-San Sebastián, September 2016

Acknowledgments

First and foremost, I thank Rodrigo Agerri for being a great supervisor, for having that much patience with a linguist stubborn enough to enter the world of programming and NLP, and for all his advice. Zu oso irakasle ona zara! No less important, I thank my second supervisor, German Rigau, for motivating me to choose this topic for my thesis.

I am also thankful to the whole IXA research group and all teachers in the LAP Master's programme who made me feel welcome from the very beginning of my stay in Donostia and who did a great job in sharing their passion for this fascinating field. Not to forget the technical staff that helped me to run my experiments on the group's computer cluster. Eskerrik asko guztioi!

If it were not for the leave of absence granted by the University of Innsbruck I would not have been able to study in the LAP programme. I am very thankful that I was given the chance to learn something new and to make all those invaluable experiences.

A very special thanks goes also to Josu – I could not have found a better flatmate. But not only him, all my other friends who I met in Donostia made this year so special.

My deepest gratitude goes, of course, also to my parents and family for all what have done for me.

Finally, I am more than grateful to my love Babsi, who encouraged me to pursue my ideas and to try something new despite not being able to be with me during that time. The support you gave me means so much to me!

Abstract

In morphologically complex languages, many high-level tasks in natural language processing rely on accurate morphosyntactic analyses of the input. However, in light of the risk of error propagation in present-day pipeline architectures for basic linguistic pre-processing, the state of the art for morphosyntactic tagging is still not satisfactory. The main obstacle here is data sparsity inherent to natural language in general and highly inflected languages in particular.

In this work, we investigate whether semi-supervised systems may alleviate the data sparsity problem. Our approach uses word clusters obtained from large amounts of unlabelled text in an unsupervised manner in order to provide a supervised probabilistic tagger with morphologically informed features. Our evaluations on a number of datasets for the Polish language suggest that this simple technique improves tagging accuracy, especially with regard to out-of-vocabulary words. This may prove useful to increase cross-domain performance of taggers, and to alleviate the dependency on large amounts of supervised training data, which is especially important from the perspective of less-resourced languages.

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The use of Natural Language Processing (NLP) technologies continues to gain importance in academic, commercial, as well as governmental settings worldwide. While NLP applications are becoming increasingly sophisticated and enable users to perform more and more complex tasks on natural language, the core of most, if not all, NLP tools consists of several steps of basic linguistic processing, including text segmentation (paragraph and sentence splitting), tokenization, part of speech (POS) tagging, lemmatisation, named entity recognition and classification (NERC), and parsing. Nowadays, these tasks are usually being performed in a modular and sequential manner. Such systems are commonly referred to as NLP pipelines.

Given the fundamental importance of those core tasks, high-quality and efficient NLP tools are indispensable to allow both researchers in academia and end-users in industry to fully exploit the possibilities of present-day NLP. From the perspective of end-users, the usefulness of a system does not only depend on performance, but also on aspects such as user-friendliness, flexibility and open availability. However, many systems are rather difficult to handle and require substantial IT skills in order to be able to embark on NLP or computational linguistics. To further advance these two fields, we believe that truly easy-to-use and readily available NLP tools that at the same time yield state-of-the-art precision and efficiency are of utmost importance. While such tools are, indeed, available for some linguistic communities, first and foremost for English (e.g. Manning et al., 2014; Padró et al., 2010) the picture is quite different for a large number of languages, especially for less-resourced languages and languages with complex morphology. As a matter of fact, we experienced this very situation on our own when trying to develop a simple application for computer-assisted learning of Polish, which required segmentation/tokenisation and morphosyntactic tagging. It turned out that the available tools needed cumbersome compilation and configuration effort as well as third-party dependencies, which means that many potential users will refrain from using them. The most promising tool in terms of user-friendliness we have found for Polish is the *PSI Toolkit* (Jassem, 2013); however, by the time we started this project only the web interface was fully functional, whereas local distributions were only partially available. Against this background, the initial motivation for the present work was to provide tools that meet the above requirements in the context of Polish. We hope that such tools will eventually spark an interest in data-driven research methodologies among the 'traditional' linguistics community and promote the development and improvement of higher-order NLP applications.

Although Polish is well equipped with large corpora, treebanks and similar resources of high quality, the very nature of the complex, inflectional morphology of Polish and many other languages poses specific difficulties to NLP tasks within the machine learning paradigm. The major problem here is data sparsity, which can be conceived as the limited availability or even complete lack of training instances for a large number of outcomes to be predicted by machine learning models. This does not imply that the training data in question is of low quality, but that natural language and thus linguistic data is inherently sparse given the open-endedness of language as a dynamic system, as already postulated by the well-known Zipf's law

(Zipf, 1935). In morphologically complex languages this data problem is even more severe. In order to perform well in real-world higher level applications, NLP pipelines for morphologically complex languages need to address sparsity issues in an adequate way. Therefore, tools for morphosyntactic tagging still deserve our attention; and to maximally comply with end-users' practical requirements in academia and industry (performance, simplicity, flexibility, cost-efficiency, minimal use of language-specific knowledge or components, etc.) such tools are ideally to be based on language-independent architectures.

1.2 Goals

In light of the motivation outlined above, we evaluate the POS tagger of the IXA pipes (Agerri, Bermudez and Rigau, 2014) when applied to morphosyntactic tagging of Polish. The decision to give IXA pipes preference over other candidate tools is mainly due to its user-friendliness: from all tools that we reviewed and tested, it convinced us most for being easy to install and use due to its readily available lightweight distribution, language-independent and trainable architecture, portability, and, no less important, for its public availability under an Apache 2.0 open-source licence. At the downside, it is not yet entirely suitable to address the problems specific to morphosyntactic tagging in highly inflected languages. Therefore, the main goal of the present thesis is to investigate to what extent semi-supervised and linguistically uninformed approaches (e.g. Spoustová et al., 2009; Turian, Ratinov, and Bengio, 2010; Eger, Gleim and Mehler, 2016) are feasible to deal with the problem of data sparsity inherent to morphosyntactic tagging of inflective languages.

Our proposed approach employs a combination of shallow local features with word clusters obtained from large amount of unlabelled text in an unsupervised manner, thus yielding a language-independent architecture that does not require language-specific feature tuning. Most importantly, we focus on the effect of including word cluster features in the morphosyntactic tagging task by conducting tests on a number of Polish datasets. This research-oriented goal aims to eventually pave the way for a truly user-friendly, simple-to-use and efficient NLP pipeline for the Polish language, which at the same time can be applied to other morphologically rich languages, too. That said, the *development* of such a full-fledged system is beyond the scope of this thesis. Instead, it is an aim to be achieved at a later stage.

The contribution of our work consists essentially in shedding light on whether unsupervised word clustering (Brown et al., 1992; Clark, 2003; Mikolov et al., 2013), a technique well known in distributional semantics, has the potential to compensate for data sparsity in supervised morphosyntactic tagging of languages with a complex morphology. Since our approach solely relies on unlabelled text rather than explicit linguistic knowledge or language-specific feature tuning, we assume that our findings are valid not only for Polish but also for other highly inflected languages. Furthermore, we also provide evidence that our approach may improve out-of-domain robustness of taggers, too. Finally, we discuss what modifications need to be implemented in IXA pipes in order to make it truly useful for high-quality morphosyntactic tagging of Polish and other morphologically rich languages.

1.3 Thesis Structure

The structure of this thesis is as follows: In Chapter 2, we describe and delimit key concepts and provide the theoretic background related to morphosyntactic tagging in morphologically complex languages. Chapter 3 summarizes the experimental setup of our study and presents the corpora and tools we employed in our research. Chapter 4 reports the results of our tests and discusses how our findings relate to the aims of our research on the one hand and to related work on morphosyntactic tagging on the other. Finally, in Chapter 5 we summarize the main findings of our study, highlight its limitations and strengths, draw conclusions and point out future lines of research and work.

2 Morphosyntactic Tagging in Highly Inflected Languages

2.1 Problem Description

The focus of our work is the basic linguistic processing task of morphosyntactic tagging, or morphological tagging, in the context of Polish as a representative of highly inflected languages. Morphosyntactic tagging is closely related to POS tagging. Both terms are sometimes even used as synonyms because the underlying problem is very similar and they can both be treated as sequence labelling problems. However, these two tasks considerably differ in complexity and therefore the two terms should clearly be kept apart, which is, unfortunately, not always the case in the relevant literature. POS tagging consists in a assigning a label for the part of speech, or word class, to each token in a sequence, for example verb, noun, adjective, punctuation mark, and so on. Morphosyntactic tagging goes further than that by assigning not only POS tags for word class – sometimes referred to as coarse grained tag or label – but also tags for morphological, or inflectional, categories, such as number, case, gender, tense, aspect, and so on. Together with POS tags, tags for inflectional categories form complex, fine-grained tags. The structure of coarse-grained POS labels is, by nature, quite different from fine-grained morphosyntactic labels. Consider Example 1, where Polish *robiła* is labelled as a past tense verb (*praet*) with the values *singular* for number, *feminine* for gender and *imperfective* for aspect, whereas its English counterpart *made* is only labelled as a past tense verb (*VBD*) according to the Penn Treebank (Marcus, Santorini and Marcinkiewicz, 1993). The internal structure of fine-grained morphosyntactic labels varies across languages, but the basic idea is always the same: using delimiter symbols (e.g. colon, plus or pipe) word class labels are concatenated with values for inflectional categories. To promote cross-language applications, there are noteworthy and highly desirable initiatives aiming to harmonise tagsets across languages (Petrov et al., 2012; Nivre et al., 2016).

(1) PL: *robiła* praet:sg:f:imperf
EN: *made* VBD

While English POS tagging is commonly considered a solved task[1] with per-token accuracies close to 98% (Toutanova et al., 2003; Yang, Salakhutdinov and Cohen, 2016), for highly inflective and agglutinative languages the situation is quite different. The reason for this is first and foremost the cardinality of the tag set: while for an isolating or analytic language such as English it is typically between 40 and 75, the average tag set found in a corpus of morphologically complex languages is between 500 and 1,000 distinct tags – whereas the amount of possible and linguistically plausible tags can reach 3,000 to 5,000 (Hajič, 2000). The problem with such large tagsets is data sparsity, i.e. the limited number or complete lack of training instances for large portions of linguistic phenomena that factually

[1] This position is becoming increasingly challenged, firstly because in real-world rather than research settings out-of-domain performance is still an issue and, secondly, because sentence accuracy is still not satisfactory despite very high per-token tagging accuracies (for a detailed discussion, see for example Giesbrecht and Evert, 2009).

occur in natural language, which decreases prediction accuracies in machine learning scenarios. The linguistic explanation for data sparsity in highly inflectional languages is that, contrary to isolating or analytic languages, they have a larger vocabulary, because inflection requires words to express certain morphological categories (e.g. tense, aspect, number, gender, etc.), mostly by affixation. The result is that on average words in inflective languages have higher morpheme per word ratios – which per se does not increase vocabulary size. More importantly, due to inflection there are more word forms per lemma than in isolating or analytic languages. Consider the following example: for the English lemma *make*, there are only four different word forms in the vocabulary: *make* (VB or VBP), *makes* (VBZ), *made* (VBD or VBN), and *making* (VBG). For the Polish lemma *robić* 'make' on the other hand there are 27 different word forms, as shown in Table 1.

It is worth noting that in both English and Polish there are ambiguities, since some word forms (e.g. *make* or *robił*) may be assigned at least two different tags if only the token itself is considered. But the real problem here is not ambiguity – in many cases, disambiguation can be achieved by positional information of tokens within a sequence – but that for a considerable number of word forms, there is only one instance in the corpus a machine learning model can learn from (see the frequency counts in in Table 1).

Word Form	Morphosyntactic Label	Count	Word Form	Morphosyntactic Label	Count
robić	inf:imperf	119	robimy	fin:pl:pri:imperf	15
robi	fin:sg:ter:imperf	159	robiona	ppas:sg:nom:f:imperf:aff	3
robią	fin:pl:ter:imperf	68	robioną	ppas:sg:acc:f:imperf:aff	2
robiąc	pcon:imperf	2	robione	ppas:pl:nom:n:imperf:aff	1
robiący	pact:pl:nom:m1:imperf:aff	1	robione	ppas:pl:nom:m3:imperf:aff	1
robicie	fin:pl:sec:imperf	5	robione	ppas:pl:acc:n:imperf:aff	1
robię	fin:sg:pri:imperf	31	robione	ppas:pl:acc:m3:imperf:aff	1
robienia	ger:sg:gen:n:imperf:aff	8	robione	ppas:sg:nom:n:imperf:aff	4
robienie	ger:sg:nom:n:imperf:aff	3	robionego	ppas:sg:gen:m3:imperf:aff	1
robieniem	ger:sg:inst:n:imperf:aff	1	robiono	imps:imperf	4
robieniu	ger:sg:loc:n:imperf:aff	2	robiony	ppas:sg:nom:m3:imperf:aff	1
robił	praet:sg:m2:imperf	1	robionych	ppas:pl:loc:f:imperf:aff	1
robił	praet:sg:m3:imperf	3	robionych	ppas:pl:gen:n:imperf:aff	1
robił	praet:sg:m1:imperf	66	robionych	ppas:pl:gen:m3:imperf:aff	1
robiła	praet:sg:f:imperf	30	robionych	ppas:pl:gen:f:imperf:aff	1
robili	praet:pl:m1:imperf	37	robisz	fin:sg:sec:imperf	33
robiło	praet:sg:n:imperf	4	rób	impt:sg:sec:imperf	8
robiły	praet:pl:n:imperf	1	róbmy	impt:pl:pri:imperf	1
robiły	praet:pl:m3:imperf	1			
robiły	praet:pl:f:imperf	2			

Table 1: Morphological analyses and frequency counts for all forms of lemma *robić* 'to make' found in the National Corpus of Polish.

While having a low number of instances of certain word forms in a corpus is certainly less than optimal, for a high-frequency verb such as *robić* 'to make' we can at least assume that there is at least one instance for each morphologically plausible

word form in a reasonably large corpus. If we take, however, a less frequent word or a term specific to a certain subject domain, chances are high that only a few of all possible word forms pertaining to a certain lemma are represented in a training corpus for machine learning. This problem of *out-of-vocabulary* words is another major obstacle in morphosyntactic tagging, because no matter how large a training corpus is, according to Zipf's law many word forms will be either completely unknown to a tagger or modelled upon a very limited number of occurrences in the training data.

Summing up, the difficulties in morphosyntactic tagging of highly inflected languages boil down to the cardinality of the set of complex, fine-grained tags; the inevitable presence of word forms unknown to a model; and multiple, ambiguous morphological tags for individual word forms.

2.2 Approaches to Morphosyntactic Tagging

The fact that both POS and morphosyntactic tagging are still a topic that receives considerable attention in NLP shows that these tasks have not been resolved yet. The reason for the continuous interest is obvious: being one of the first preprocessing steps to be carried out in sequential and ascending NLP pipelines, high quality on this lower level of language processing is crucial to the successful application of high-level and more complex tasks. Because of the risk of error propagation throughout NLP pipelines, insufficient performance on lower levels may seriously affect performance of high-level tasks (Caselli et al., 2015).[2] Current lines of research in morphosyntactic tagging include the following: 1) improving the overall state of the art, especially for inflective and agglutinative languages as well as less-resourced languages; 2) improving sentence accuracy; 3) improving cross- and out-of-domain robustness (Giesbrecht and Evert, 2009); 4) unsupervised induction; and 5) improving the taxonomic basis of the linguistic resources for training data within the framework of descriptive linguistics (Manning 2011).

For POS tagging, the most commonly applied machine learning algorithms are Maximum Entropy (Ratnaparkhi 1996; Toutanova et al., 2003), Perceptron (Collins, 2002), Hidden Markov Models (Brants, 2000), Support Vector Machines (Giménez and Màrquez, 2004), and most recently neural networks (Yang, Salakhutdinov, and Cohen, 2016). These algorithms are being successfully applied to a variety of languages. To yield state-of-the-art results on per-token accuracy, for many languages it is enough to train models with these algorithms using specific feature sets, without combining taggers with any other component.

For morphosyntactic tagging, the approaches are not as straight-forward as in the case of POS tagging due to the difficulties outlined in section 2.1. One may assume that a simple solution to alleviate the data sparsity problem would be to reduce the tagset by stripping fine-grained tags, thus keeping only coarse-grained POS labels. While for some tasks this may be a feasible solution, for most high-level applications this would imply an unwarranted deprivation of important linguistic information. After all, in highly inflective language word order is fairly flexible as compared to isolating or analytic languages, which means that syntactic infor-

[2] Whether pipelining is the best choice for NLP is a research question in its own right and therefore beyond the scope of this thesis. Without doubt, it is still a dominant and influential approach. This may change with deep learning and neural network approaches that increasingly gain momentum.

mation is primarily encoded through morphological categories and agreement between word forms. Or, as Hajič (2000, p. 94) described it succinctly:

> "These languages, obviously, do not use the rich inflection just for the amusement (or embarrassment) of their speakers (or NLP researchers): the inflectional categories carry important information which ought to be known at a later time (e.g., during parsing). Thus one wants not only to tell apart verbs from nouns, but also nominative from genitive, masculine animate from inanimate, singular from plural - all of them being often ambiguous one way or the other."

To ensure its usefulness for high-level tasks, morphosyntactic tagging should therefore retain as much of the morphologically encoded information as possible. Since supervised machine learning is, by nature, not suitable to learn low-frequency instances, they may be excluded from training data, or their fine-grained tags may be replaced with coarse-grained ones; this requires additional techniques, based, for example, on rules or dictionaries. In any case, to adequately address the data sparsity problem caused by low-frequency instances, supervised machine learning requires complementary approaches. A review of existing systems reveals that the following approaches can be distinguished:

- **Morphological analysis:** With out-of-vocabulary words being the major obstacle to high accuracy, a reduction of unknown words can improve tagging greatly. This can be achieved by means of a morphological analyzer that provides the tagger itself with a list of all possible tags for each token, which basically converts the tagging task into a disambiguation task. For morphological analysis, two different techniques are being commonly applied: Firstly, using a lexicon of inflectional forms (e.g. Georgiev et al., 2012 for Bulgarian or Woliński et al., 2012 for Polish). Secondly, applying a morphological guesser rather than a static morphological lexicon. Guessers aim to dynamically infer all possible labels for each token in test time, especially for unknown words, mainly based on pre- and suffix analysis. In fact, many, if not most, morphologically complex languages heavily rely on guessers, for example Icelandic (Loftsson and Östling, 2012), Czech (Straka et al., 2016), Hungarian (Oravecz and Dienes, 2002), or Polish (Radziszewski, 2013).
- **Tiered Tagging:** Tiered tagging essentially consists in treating complex morphosyntactic tags not atomically but each sub-label representing a certain inflectional value separately in a sequential manner. Thereby, the number of tiers, or layers, corresponds to the number of inflectional categories distinguished. It has been used, among others, for Romanian (Ceauşu, 2006), Hungarian (Tufiş, 1999; Tufiş and Dragomirescu, 2004), or Polish (Radziszewski, 2013; Radziszewski and Śniatowski, 2011).
- **More Complex Models:** This approach refers to training more sophisticated machine learning models than the ones described for POS tagging. Here, first and foremost Conditional Random Fields (CRF) are being applied. CRFs are a class of probabilistic models for structured prediction that can be successfully applied to sequence labelling problems with a large number of interdependent variables (Sutton and McCallum, 2011). Although model complexity and training time are usually much higher than in the case of simpler algorithms, CRF have become quite popular for morphosyntactic tagging thanks to the high accuracy it yields (e.g. Radziszewski, 2013;

Silfverberg et al., 2014). Müller, Schmid and Schütze (2013) have proposed a CRF tagger that achieves state-of-the-art scores across six languages while at the same time being fast enough to deal with large training corpora in reasonable time, what is otherwise a major limitation of CRF.

- **Ensemble Voting:** In ensemble systems, several individual taggers are combined in a complementary way, such that a voting technique is applied to choose the most probable prediction from each of the individual taggers. For Polish, this approach has achieved state-of-the-art results (Kobyliński, 2013; 2014; Radziszewski and Śniatkowski, 2011).
- **Semi-supervised leveraging of unlabelled data**: There have been promising approaches to combine supervised tagging with unsupervised techniques for the exploitation of large amounts of unlabelled data, thus resulting in semi-supervised approaches. Successful applications include Spoustová et al. (2009) or Eger, Gleim and Mehler (2016), who use word embeddings. Such approaches are not only suitable for POS/morphosyntactic tagging, but generally for a wide range of sequence labelling problems. Thus, Agerri and Rigau (2016) have demonstrated significant improvements in NERC, while Yang, Salakhutdinov and Cohen (2016) leverage word embeddings from large corpora to score state-of-the art results on several benchmark tasks across languages, including POS tagging and NERC.

Of course, these approaches are not mutually exclusive, meaning that taggers may exhibit characteristics of more than one approach.

3 Materials and Methods

In this chapter, we describe the experimental setup of our study (section 3.1), the architecture of our morphosyntactic tagger (section 3.2), as well as the corpora used for the supervised (section 3.3) and unsupervised (section 3.4) components of our system. Our tagger uses *ixa-pipe-ml*, the centralised machine learning module of the IXA pipes.

3.1 Experimental Setup

Our research included the following steps:

1. Running *ixa-pipe-ml* off the shelf on Polish data, i.e. without any modifications to the source code or feature set. This preliminary step corroborated our initial assumption that in its current form, the IXA pipes are not suitable for precise morphosyntactic tagging in morphologically rich languages. Yielding per-token accuracies of approximately 86%, results were well below the current state of the art for Polish and other highly inflected languages.
2. In a further preliminary step we examined whether models trained with CRF are a viable solution to achieve our goals. We opted for CRF because systems based on this algorithm are among the top-scoring systems for a number of morphologically complex languages, including Polish (Radziszewski, 2013). To test this approach, we used the *mallet addon*[3] for OpenNLP[4] to implement the CRF algorithm in *ixa-pipe-ml*. The add-on allows to train CRF models in OpenNLP-based systems using the API of the MALLET machine learning library (McCallum, 2002). However, after running one experiment with our implementation – the code is made available on GitHub[5] – we decided to discard and to no longer pursue this avenue. The decision is based on the fact that while improving tagging accuracy by almost 3.5% as compared to Perceptron models, CRF training is unbearably slow. Thus, it took 32 days to train a model for a training set of 69,499 tokens, as opposed to only three minutes with the Perceptron algorithm. CRF models can be very complex, their training is computationally costly and they are said to require extensive feature engineering (Baldwin, 2006); therefore we consider that this approach does not fully meet the requirement of simplicity and efficiency of the IXA pipes.
3. Following the methodology applied by Agerri and Rigau (2016) to NERC, we established our baseline system, on top of which clustering features are to be added. This step was done by running several hundred tests with *ixa-pipe-ml* in order to find the best configuration of the local feature set, where 'local' refers to shallow non-clustering features, such as the token itself, word shape or *n*-grams. Apart from the local features already supported by *ixa-pipe-ml*, we implemented a small number of additional local features, which were reported to be useful in other systems identified from the literature. Details on our baseline feature set, as well as tagging scores obtained from it, are given in section 3.2.2.

[3] http://svn.apache.org/viewvc/opennlp/sandbox/mallet-addon
[4] https://opennlp.apache.org/index.html
[5] https://github.com/mustaszewski/ixa-pipe-pos/tree/CRFTrainer

4. Distributional cluster lexica to be used as clustering features atop of the baseline were induced from four different unlabelled text collections (see 3.2.3).
5. Based on the two strongest baselines – one for the Perceptron and one for the Maximum Entropy training algorithm, respectively – another series of several hundred tests was run on the development data sets to determine which (combinations of) clustering features improve the baselines most.
6. Once the best combinations of local features and clustering features were identified, final evaluations were performed on the chosen test sets.

Throughout our experiments, we trained both Perceptron and Maximum Entropy models in order to compare the performance of these two algorithms implemented in *ixa-pipe-ml*. Consequently, all results are reported for both models in the remainder of this thesis.

Given that except for the first step we did not use the IXA pipes off the shelf but implemented a number of modifications, we make publicly available[6] the source code of the system in order to guarantee reproducibility of our results. Our commits mainly concern the addition of features specific to morphosyntactic tagging.

3.2 System Description

3.2.1 IXA Pipes

For our experiments, we use the IXA pipes, which are a set of modular NLP tools that have been developed to lower the barriers of using NLP technology while at the same time yielding state-of-the-art results (Agerri, Bermudez and Rigau, 2014). The IXA pipes are a simple and ready-to-use yet efficient toolkit for basic linguistic preprocessing and annotations: sentence segmentation, tokenization, POS tagging, NERC, chunking, and constituent parsing. These tasks are performed by a series of modules: *ixa-pipe-tok*, *ixa-pipe-pos*, *ixa-pipe-nerc*, *ixa-pipe-parse*, and *ixa-pipe-chunk*. The IXA pipes have a language-independent, multilingual architecture, which means that they can be trained and used with any language without language-specific parameter tuning. They are distributed under the open-source Apache 2.0 License that facilitates source code use, distribution and integration, also for commercial purposes. All modules of IXA pipes are based on the machine learning API of the Apache OpenNLP project.

In our work, we used only the centralised machine learning module *ixa-pipe-ml* because it offers a very simple way to training and evaluation parameter setting. While in earlier versions of the IXA pipes a number of features were hard-coded and thus not configurable without time-consuming source code modifications and compiling, in the current version (0.0.1), all supported parameters can be specified in a separate training parameter configuration file that is being parsed upon launching the training procedure. Hence, it provides a simple and time-saving way to performance evaluation. As a matter of fact, being the centralised machine learning component of the entire pipeline, *ixa-pipe-ml* does *not* use parameters specific to one of the remaining modules; rather, in the training parameter configu-

[6] The source code can be found in the branch *pos-pl* our fork of the IXA pipes project, see https://github.com/mustaszewski/ixa-pipe-ml/tree/pos-pl

ration file users can choose from all implemented features in order to train models that are most suitable to the task in question.

3.2.2 Baseline Features

As outlined in section 3.1, the third step of our experimental workflow consisted in establishing the baseline that can be obtained from local features only, i.e. without the use of clustering features. On top of this baseline, clustering features are to be added in a subsequent step. In the following, we briefly outline the local features supported by and used in our system. To illustrate the features, we use the Polish sentence *Jest to trudne pytanie* 'This is a difficult question' as a running example. Assuming that the current token to be tagged is *trudne* 'difficult', as indicated by the arrow pointer in Table 2, the position markers from -2 to +2 identify the position of each token in the sequence relative to the token at position 0, i.e. the token currently being tagged. Of course, the tagger has access to the entire sequence of morphosyntactic tags only during training, while during testing it is the very aim of the tagger to determine the correct tag for each token of the sequence.

			↓		
Position	-2	-1	0	+1	+2
Token	Jest	to	trudne	pytanie	.
Tag	fin:sg:ter:imperf	pred	adj:sg:nom:n:pos	subst:sg:acc:n	interp
Token Class	single capital	lower	lower	lower	other
Token Shape	Xx*	x*	x*	x*	.
Sentence Begin	true	false	false	false	false
Sentence End	false	false	false	false	true
Prefix (1-3)	J, Je, Jes	t, to	t, tr, tru	p, py, pyt	.
Suffix (1-4)	t, st, est, Jest	o, to	e, ne, dne, udne	e, ie, nie, anie	.

Table 2: Running example of tagged sentence (first 3 lines) and features generated from it.

Token Features

- **Token:** The token itself at the current position to be predicted. If this feature is selected, the token can be either lowercased or retained in its original form for feature generation. In our running example, the token feature is simply the current word itself, i.e. *trudne*.
- **Token Class:** Assigns a class to the token according to its shape, i.e. the characters contained. The classes are: 1) lowercase alphabetical characters only, 2) alpha-numeric characters, 3) digits only, 4) single capital letter only, 5) capital letters only, 6) one capital letter followed by a period, and 7) other. In Table 2, token class features for each word of our running example are shown. In addition, this feature can be combined with the token itself (lowercased or original), yielding joint features of the form *token, class*, for example *Jest*, *single capital* or *trudne*, *lower*.
- **Token Shape:** Following Ciaramita and Altun (2006), this feature normalises each token by substituting individual characters or sequences of characters according to whether they are uppercase, lowercase, digits or other characters. In Table 2, the normalisation was performed for each token of the running example.

Previous Outcome Features

- **Previous prediction:** If the token to be predicted at the current position has already appeared previously in the data, the previous tagging decision is retrieved.
- **Preceding outcomes:** Uses the tags assigned to tokens preceding the current token. A separate parameter can be used to specify how many preceding outcomes are to be included. In our example, the two preceding outcomes are *fin:sg:ter:imperf* at position -2 and *pred* at position -1. Furthermore, an *n*-gram of preceding tokens can be used for feature generation, with the size of the *n*-gram to be specified by the user, for instance the bigram *fin:sg:ter:imperf*, *pred* in our running example. Also, joint features of the preceding outcome with the token itself (e.g. *pred*, *trudne*) as well as the previous outcome with token class (e.g. *pred, lower*) can be generated.
- **Preceding Sub-Label Features:** Inspired by the feature templates employed by Radziszewski (2013) and Silfvereberg et al. (2014), we implemented this feature which does not treat previous outcome labels atomically, but extracts specific inflectional values from previous labels. For example, values for the morphological categories of case, gender, number or aspect of the preceding outcomes in a specified range can be used for feature generation. To this end, in the training configuration file the following parameters need to be specified: a) which character is used as separator to delimit sub-label values; b) the grammatical class(es) of interest, followed by the set of all possible values for the respective class(es). For instance, specifying the colon as separator and providing the sets of possible values for the classes *number={sg|pl}* and *case={nom|gen|dat|acc|inst|loc|voc}* will extract the values *number=sg* and *case=acc* from the compound label *subst:sg:acc:n*, whereas from the label *fin:sg:ter:imperf* only the value *number=sg* will be used for feature generation. Furthermore, it can be specified whether the word class, i.e. the coarse-grained POS tag, is to be extracted from the compound label. In the case of our running example, this would be *pred* at position -1 and *fin* at position -2. With this feature, the intuition is that sub-label dependencies help to capture morphological agreement, at least to a certain extent.

Sentence Features

- **Sentence beginning**: A binary feature assigning *true* to the first token of a sentence, as shown in Table 2.
- **Sentence end**: A binary feature assigning *true* to the last token of a sentence, as shown in Table 2.

Pre- and Suffix Features

- **Prefix**: Gets the prefixes of the current token, with the minimum and maximum prefix length to be specified. In Table 2, all possible prefixes of length one to three for each token of the running example are shown.
- **Suffix**: Gets the suffixes of the current token, with the minimum and maximum suffix length to be specified. In Table 2, all possible suffixes of length one to four for each token of the running example are shown.

N-Gram Features

- **Token Class N-Grams**: Generates token class *n*-grams in a specified range before and after the current token, where *n*-grams can range from bi- to fivegrams. For instance, the token class bigram preceding the current token is *single capital, lower*, while the following bigram is *lower, other*.
- **Character N-Gram Features:** Generates features from all character *n*-grams up to a specified length to be found in the current token. For example, all possible character trigrams of the current token *trudne* are *tru*, *rud*, *udn*, and *dne*.

For token and token shape features, window size needs to be specified, i.e. the range of tokens preceding and following the current token to be considered for feature generation. For instance, setting window range to 2:2 will generate token features from the current token (*trudne* in our running example), the two preceding tokens (*Jest*, *to*), and the next two tokens (*pytanie, .*), resulting in a window of size 5.

In Table 3, the training parameter configurations for our two best-performing baselines (for Perceptron and Maximum Entropy models, respectively) are provided by specifying the precise setting for each of the above mentioned features. Tagging accuracies for our baselines are reported in Chapter 4.

Feature	**Settings**	
	Maximum Entropy	**Perceptron**
Window size	1:1	2:2
Token	yes	yes
Lowercase token?	yes	yes
Token class	no	yes
Token and class?	---	yes
Token shape	no	yes
Preceding outcomes	yes	no
Preceding outcomes range	-2	---
Preceding outcome *n*-grams	no	no
Previous Prediction	yes	yes
Sentence beginning	yes	yes
Sentence end	yes	true
Prefix length min	1	1
Prefix length max	2	2
Suffix length min	1	1
Suffix length max	5	4
2-gram class	no	yes
3-gram class	no	yes
4-gram class	no	no
5-gram class	no	no
Character n-grams range	2:5	2:5
Preceding sub-labels	yes	yes
Preceding word class?	yes	yes
Preceding sub-label classes	number, gender, case	number, gender, case
Preceding sub-label range	-1	-2

Table 3: Feature configuration of our two baseline systems.

3.2.3 Clustering Features

After establishing the baselines, we induced three types of word clusters from four different corpora consisting of large amounts of unlabelled Polish text (see 3.4). The clusters, or cluster lexica to be more precise, are then used in our system to generate features by looking up to which cluster a given token belongs. The underlying idea is that all words that belong to the same cluster share some semantic or morphological information. In this way, our system exploits distributional word representations, i.e. the clusters, in a semi-supervised manner: While training of Perceptron and Maximum Entropy models is a supervised task based on annotated corpora, the use of word representations obtained from unlabelled corpora based on distributional information is an unsupervised task. The methodology of combining supervised and unsupervised techniques in NERC has been described in detail by Agerri and Rigau (2016).

The three types of word clusters used in our system are: Brown clusters, Clark clusters and K-means clusters based on the skip-gram algorithm used in the word2vec tool. For NERC, it has been shown that a combination of cluster types helps to improve performance greatly, as compared to the use of individual cluster features (Agerri and Rigau, 2016).

Brown Clusters

Brown clustering (Brown et al., 1992), or IBM clustering, is used to group words that appear in similar contexts into hierarchical clusters and was originally designed to address data sparsity in *n*-gram language modelling. For the induction of Brown clusters, we use the C++ implementation of the Brown word clustering algorithm proposed by Liang (2005)[7] off-the-shelf and with default settings.

As input, the tool takes a corpus of tokens separated by whitespaces without punctuation, with one sentence per line. Following previous work (Liang, 2005), all sentences with less than 90% lowercase characters have been removed from the corpus. As output, the tool creates a hierarchical clustering lexicon, where each word is identified by a bit string that represents the word's path from the root in a binary tree. A sample input and output file extract is shown in Table 4. The features generated by our system retrieve the path for the current token from the Brown clustering lexicon, which is stored in plain text format.

Brown Cluster Input	**Brown Cluster Output**		
ten film zmusza do myślenia	1001	w	986574
czym przyciągnąć gości	01011100	i	574255
...	011100	się	2164800
	11111111010	robiła	5
	...		
	<cluster path as a bit string> <word> <frequency>		

Table 4: Sample input and output for Brown cluster induction.

Clark Clusters

Clark clusters (Clark, 2003) were originally designed to exploit distributional and morphological information to improve unsupervised POS tagging across languages,

[7] https://github.com/percyliang/brown-cluster

with a focus on rare words. Therefore, the algorithm relies only to a limited degree on frequency information. Instead, it considers sequences of letters that form each word, such that morphologically similar words are placed in the same cluster. This approach makes it a promising candidate to deal with highly inflected languages.

We induce Clark clusters using Clark's original implementation off-the-shelf. The input to the tool is a corpus of lowercase tokens, with one token per line and a blank line between sentences, while the output is a plain text file where each line represents a word type with its cluster number and a weight (see Table 5). For feature generation, our system performs simple look-ups in the cluster dictionary to retrieve the cluster number for the current token, assigning null if no entry is found in the lexicon.

Clark Cluster Input	**Clark Cluster Output**
ten film zmusza do myślenia czym przyciągnąć gości ...	w 82 0.967019 i 75 0.726565 się 25 0.985357 robiła 12 8.46919e-05 … <word> <cluster> <weight>

Table 5: Sample input and output for Clark cluster induction.

word2vec Clusters

Finally, our word2vec features are based on skip-gram embeddings clustered via K-Means (Mikolov et al., 2013). We induce the clusters using the *word2vec* tool[8] off-the-shelf, which takes a corpus of lowercased, space-separated tokens without punctuation as an input to produce a cluster lexicon, where each line represents a word type and its corresponding cluster number (see Table 6). In our system, we use word2vec clusters in the same way as Clark clusters, which means that for the current token to be predicted its cluster number is retrieved from the cluster lexicon.

word2vec Cluster Input	**word2vec Cluster Output**
ten film zmusza do myślenia czym przyciągnąć gości ...	w 37 i 37 się 20 robiła 70 … <word> <cluster>

Table 6: Sample input and output for word2vec cluster induction.

3.3 Data for Training and Testing

For training and testing, we use a total of three different, publicly available corpora: 1) the National Corpus of Polish, or *Narodowy Korpus Języka Polskiego* in Polish

[8] https://code.google.com/archive/p/word2vec/

(henceforth NCP); 2) the Universal Dependencies Polish Treebank (UDP); and 3) the Polish Language of the 1960s Corpus (P60). The first two were used for both training and testing, whereas the third one was only used for out-of-domain evaluation of the models trained on the former two corpora. Details on each of the datasets are given in the following subsections and summarised in Table 7.

Dataset	Tokens	Vocabulary (Types)	Vocabulary (Lemmas)
NCP Train	962,219	124,049	47,687
NCP Dev	119,704	31,343	15,475
NCP Test	120,949	32,339	16,150
NCP Total	1,202,872	--	--
UD Train	69,499	22,594	11,710
UD Dev	6,887	3,467	2,510
UD Test	7,185	3,558	2,554
UD Total	83,571	--	--
P60 Essays (UD)	7,632	--	--
P60 Fiction (UD)	7,448	--	--
P60 News (UD)	7,612	--	--
P60 Plays (UD)	7,037	--	--
P60 Science (UD)	7,632	--	--
P60 Essays (NCP)	120,100	--	--
P60 Fiction (NCP)	120,725	--	--
P60 News (NCP)	116,807	--	--
P60 Plays (NCP)	114,613	--	--
P60 Science (NCP)	118,505	--	--

Table 7: Statistics for datasets used for training and testing. P60 sub-corpora for out-of-domain evaluation have been compiled for testing with both NCP models and UDP models.

3.3.1 National Corpus of Polish

Being a carefully crafted, balanced corpus, the NCP (Przepiórkowski et al., 2012) has become the most influential resource for Polish NLP. This reference corpus contains more than 3,000 texts totalling 1.2 million tokens and contains manual annotations on various linguistic levels, including morphosyntactic annotation. It is distributed under a GNU GPL licence in TEI-compliant XML format.

The *ixa-pipe-ml* training component of our system takes plain text files as an input, where each line represents one token with its morphosyntactic label separated by a tabulator. To preprocess the corpus according to the required input format, we first converted all TEI file to XCES format using a Python script provided in the repository[9] of the PANTERA tagger. Subsequently, we used our own script[10] to convert XCES files into the required format. Once converted, we shuffled the order of the plain text files extracted from NCP in order to create three randomised partitions: an 80% training set, a 10% development set, and a 10% test set. To ensure reproducibility, we make our partitions publicly available[11].

[9] https://github.com/accek/pantera-tagger/blob/master/scripts/tei2xces.py
[10] https://github.com/mustaszewski/preprocessing
[11] https://github.com/mustaszewski/nkjp

We have decided to use the NCP in order to facilitate the comparison of our system to state-of-the-art taggers for Polish, since it is the most commonly used corpus in Polish NLP. Corpus statistics are given in Table 7.

3.3.2 Universal Dependencies Polish Treebank

The Universal Dependencies (UD) Project (Nivre et al., 2016) aims to develop treebank annotations consistent across many languages. From the perspective of POS and morphosyntactic tagging, an interesting feature of the UD project is that it draws upon the Google universal POS tagset (Petrov et al., 2012), thus contributing to a harmonisation of efforts across languages and research communities. Polish features among the 37 treebanks currently included in the project. For each language, two different annotation schemas are available: the universal POS tags for word class information and the extended tags (XPOS) for full morphosyntactic tags. In our work, we used only full morphosyntactic tags.

The UDP (Wróblewska and Przepiórkowski, 2014) is based on the Polish dependency treebank (Woliński, Głowińska and Świdziński, 2011), which, in turn, is based on the NCP. Therefore, UDP is a subset of NCP. Nevertheless, we decided to include UDP into our study, too, for the following reasons: Firstly, it is only approximately 7% the size of NCP; as a relatively small corpus it therefore allows us to investigate the performance of our tagger in a setting where data sparsity is even more of an issue than with large corpora. In addition, working with smaller corpora even though a large reference corpus is available may have important implications for the feasibility of our system in domains where less training data is available, or for other, potentially less-resourced languages. Secondly, all UD treebanks provide standardised training, development and test sets, which facilitate performance comparisons across systems. Thirdly, given its relatively small size and consequently short training-testing cycles, UD datasets are very handy during system development while at the same time yielding representative results. Corpus statistics are given in Table 7.

3.3.3 Polish Language of the 1960s Corpus

Given that tagging accuracy has been shown to drop considerably when applied to domains or text types different from those represented in training data (Giesbrecht and Evert, 2009), out-of-domain evaluations of new systems and approaches are of utmost importance. After all, in view of the high costs for the creation of hand-annotated data, in real-world scenarios many, if not most taggers are likely to be applied to subject domains or text types that are potentially underrepresented in training data. Consequently, robustness in cross-domain tagging is a cornerstone of systems' feasibility.

For our out of domain evaluation we use the P60 corpus (Ogrodniczuk, 2003). It contains 10,000 samples from five domains: essays, news, scientific texts, fiction and plays, all of them written between 1963 and 1967. We have chosen this corpus for out-of-domain evaluation because of the following reasons: Firstly, its tagset is fully compatible with both NCP and UDP tagsets – which is a prerequisite to test models across datasets. Secondly, contrary to many other corpora which would have otherwise been good candidates for evaluation, morphosyntactic information in the corpus is hand-annotated. Thirdly, the corpus is freely available under a GNU licence; and, fourthly, source files are neatly grouped into the five domains men-

tioned above, which makes it very handy to create domain-specific sub-corpora. That said, the P60 corpus has one important drawback: its representativeness for present-day Polish may be deemed unsatisfactory, especially with regard to language use in online and electronic communication. Nevertheless, we believe that our choice constitutes a reasonable yet not ideal trade-off between practicality and validity.

To obtain in-domain test sets for each of the five subdomains, the order of texts in the respective subdomains has been randomised and subsequently partitions, corresponding in size to the test set used in combination with the respective training data, were extracted. Thus, we compiled five out-of-domain test sets for both NCP-models and UDP-models, with sizes similar to the non-out-of-domain test sets. Corpus statistics are given in Table 7.

As outlined in section 2.1, data sparsity is one of the main obstacles to high-quality morphosyntactic tagging in highly inflected languages. By means of a simple frequency analysis, we examined if and to what extent sparsity is, in fact, a factor in Polish language data. To this end, we quantified sparsity in terms of the frequency of word forms, lemmas and morphosyntactic labels in the datasets used in our study, namely NCP and UDP.

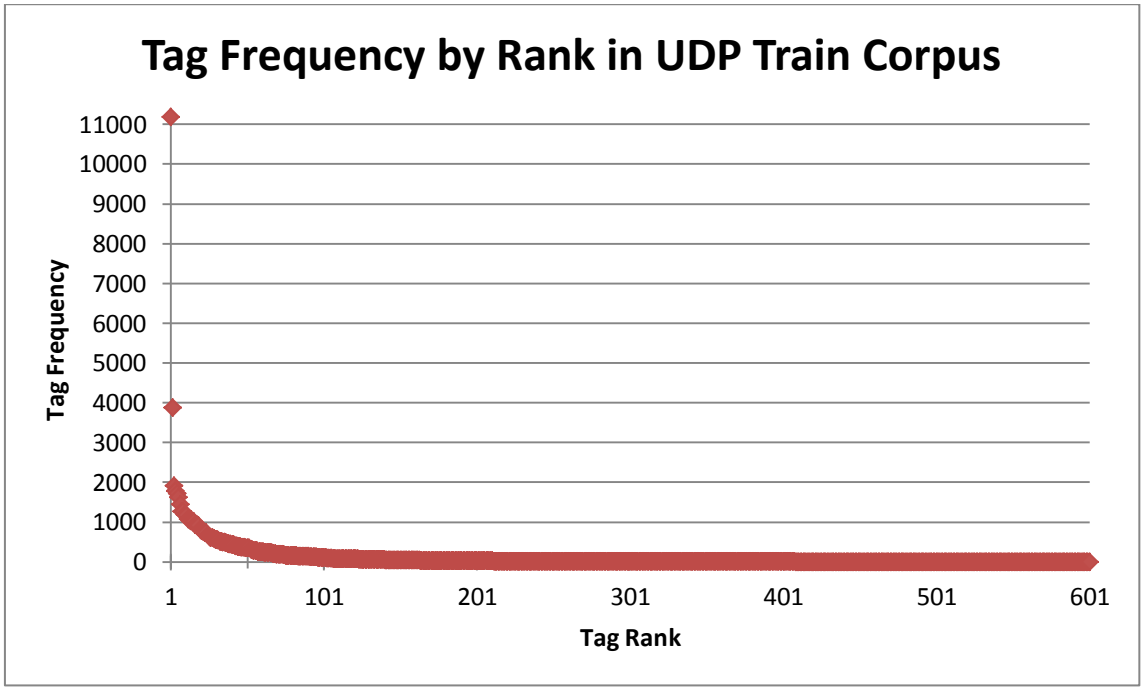

Figure 1: Frequency of labels in UD Train, sorted by rank.

Figure 1 shows the distribution of morphosyntactic labels according to their frequency rank in the corpus. The most frequent label in UDP (*interp*, not indicated in figure) occurs 11,184 times, thus accounting for 16.1% of all 69,499 tokens in the corpus, whereas the second most frequent label, *qub*, accounts for 5.6%. Taken together, the top-five labels account for 29.5%, and the top-ten for 39.2% of the data. Sparsity becomes even more obvious when considering that the 100 most frequent labels account for 89.6% of all instances, which means, in turn, that the remaining 507 labels make up only 10.4% of training data. In such a scenario, sparsity is an undeniable fact. Similarly, Table 8 provides further evidence for sparsity in the training data: For example, in NCP Train there is a total of 962.219

tokens, which are distributed among 124,049 different word forms (types). A closer look on the distribution of word forms reveals that out of these 124,049 forms, 71,793 appear only once in the corpus. In other words, for 71,793 word forms, or almost 58% of the entire vocabulary, there is only one single instance in the corpus, which makes machine learning very challenging. For 115,660 word forms, or an overwhelming 93.2%, there are only ten or less instances the model can learn from. The picture is similar yet not as extreme if we look on the tags in the corpus: The tagset consists of a total of 901 different tags, and for 116 of them, or 12.9%, there is only one training instance. All these figures are clear indicators of data sparsity.

	Tokens	**Lemmas**	**Word forms**				**Tags**			
			total	**1x**	**≤ 5**	**≤ 10**	**total**	**1x**	**≤ 5**	**≤ 10**
NCP Train	962,219	47,687	124,049	71,793	108,135	115,660	901	116	236	305
NCP Dev	119,704	15,475	31,343	21,52	29,376	30,402	665	102	247	326
NCP Test	120,949	16,15	32,339	22,451	30,403	31,465	667	93	237	309
UDP Train	69,499	11,710	22,594	16,442	21,447	22,088	607	102	252	325
UDP Dev	6,887	251	3,467	291	3,377	3,422	333	93	196	232
UDP Test	7,185	2,554	3,558	2,952	3,462	351	349	107	213	250

Table 8: Sparsity in training and evaluation data.

3.4 Data for Unsupervised Word Cluster Induction

Contrary to training and evaluation of our morphological tagger, the induction of word clusters to be subsequently used as features in our system does not require labelled, i.e. supervised, data. Instead, the algorithms induce word clusters (for a brief description of the cluster types used see 3.2.3) from unlabelled data in a purely unsupervised fashion, which is ideally done on large corpora. In the following, we briefly describe the four corpora used for cluster induction.

Araneum Polonicum Maius

The *Aranea* project comprises comparable Gigaword corpora obtained by web crawling in a number of languages (Benko, 2014; 2016). For each language, a large version of approximately 1.2 billion tokens and a smaller, randomly sampled subset of approximately 120 million tokens are available. For our system, we used the *Araneum Polonicum Maius* of 1,110,120,694 tokens.[12] Although the distributed version contains automatically generated morphosyntactic annotation and lemmas, we were only interested in extracting plain text.

Polish Wikipedia

The Wikipedia, being one of the largest freely available text sources, has proved useful for cluster induction, too. We downloaded the most recent dump and after text extraction from the XML files and subsequent corpus preprocessing we obtained approximately 185 million tokens.

[12] We would like to thank Vladimír Benko for granting us access to the corpus.

Polish Sejm Corpus

The Polish Sejm Corpus (PSC) is a collection of stenographic transcripts from sittings of the Polish Sejm, totalling approx. 200 million tokens (Ogrodniczuk, 2012). It contains automatically generated annotation on various levels encoded in TEI format. However, we extracted only plain text using our own scripts[13]. After preprocessing, we obtained approximately 165 million tokens. In large parts, the transcripts represent quasi-spoken speech, which makes it a very interesting resource.

PELCRA

Finally, the smallest of our unlabelled corpora is the PELCRA multilingual parallel corpus of Polish[14], from which we extracted all Polish source texts, leaving us with 35 million words. The corpus comprises texts related to the European Union: news articles from the web page of the Community Research and Development Information Service, press releases from the European Commission, the European Parliament and the European Southern Observatory.

All four corpora have been preprocessed according to the specifications of each cluster induction tool. Preprocessing, as well as cluster induction, was carried out on the computer cluster of the IXA NLP group at the University of the Basque Country because the large amount of data can hardly be handled by most standard personal computers. For each cluster type and each corpus, a number of clusters with different amounts of cluster classes have been induced, 42 in total (see Table 9). Induction time varied greatly across corpora and cluster types, ranging from 1.5 hours to almost one month. Whereas Brown and especially word2vec clusters can be induced in relatively short amount of time thanks to multithreading, the implementation for Clark clusters supports single-threaded induction only, which leads to very long training times, especially in the case of Gigacorpora.

[13] https://github.com/mustaszewski/preprocessing
[14] http://pelcra.pl/new/multilingual

Type	Corpus	No. of Classes
Brown	Araneum	500
Brown	Araneum	1000
Brown	Araneum	2000
Brown	Pelcra	500
Brown	Pelcra	1000
Brown	Pelcra	2000
Brown	Sejm	500
Brown	Sejm	1000
Brown	Sejm	2000
Brown	Wiki	500
Brown	Wiki	1000
Brown	Wiki	2000
Brown	Wiki	3200
Clark	Araneum	100
Clark	Araneum	200
Clark	Pelcra	100
Clark	Pelcra	200
Clark	Pelcra	400
Clark	Sejm	100
Clark	Sejm	200
Clark	Sejm	400
Clark	Sejm	600
w2v	Araneum	100
w2v	Araneum	200
w2v	Araneum	400
w2v	Araneum	800
w2v	Araneum	1000
w2v	Pelcra	100
w2v	Pelcra	200
w2v	Pelcra	400
w2v	Pelcra	800
w2v	Pelcra	1000
w2v	Sejm	100
w2v	Sejm	200
w2v	Sejm	400
w2v	Sejm	800
w2v	Sejm	1000
w2v	Wiki	100
w2v	Wiki	200
w2v	Wiki	400
w2v	Wiki	800
w2v	Wiki	1000

Table 9: Overview of induced clusters.

3.5 Evaluation Metrics

Over the last few years, in the Polish NLP community there have been different approaches to measuring tagging accuracy, which, unfortunately, makes comparisons across systems difficult. The use of different metrics reflects different conceptions as to what the exact task of a tagger is. According to Radziszewski and Acedański (2012), there are three answers to this question:

1. Taggers tag plain, i.e. non-segmented and non-tokenised, text.
2. Taggers tag a sequence of unlabelled tokens rather than unlabelled plain text. This assumes that tagging is performed on perfectly segmented and tokenised input, which is not always the case in real-world applications.
3. The tagger's task is to disambiguate the correct tag among a set of possible tags previously obtained from dictionary lookups or other techniques for morphological analysis.

These three conceptions led to the use of various metrics as the statistic figure to be reported in papers:

- **Disambiguation accuracy**: This statistic figure is related to the third of the above conceptions. It used to be the most popular one for Polish, despite being the most controversial one because of relying on gold-standard morphological analyses from the reference corpus, which largely neglects real-world scenarios.
- **Per-token accuracy,** also called **word accuracy**, is associated with the second conception. This metric indicates what percentage of test tokens has been assigned the correct tag by the system. It is the standard metric worldwide. To better capture taggers' performance, especially in highly inflected languages, it is common practice to report accuracy scores for words known to the model (i.e. words that did appear in the training data) and unknown to the model words (i.e. words that did *not* appear in the training data, also called out-of-vocabulary words) rather than calculating only overall word accuracy in an undifferentiated manner.
- **Accuracy lower bound**, finally, reflects the first of the three conceptions. This statistic figure, suggested by Radziszewski and Acedański (2012), counts the number of output tokens that have been correctly segmented *and* are labelled correctly according to the gold standard, divided by the total number of tokens in the reference corpus. In this metric, differences in tokenisation are penalised, i.e. tokens from the reference corpus that are not identically present in the tagger output are treated as a tagging error. Apart from that, they also recommend to report **accuracy upper bound**, which is the hypothetical upper limit of tagger performance "under the (false) assumption that each token from the reference corpus that was not explicitly present in tagger output due to unexpected tokenisation would be correctly tagged" (Radziszewski, 2013, p. 252). This two metrics have become the standard in Polish NLP.

The difficulty when comparing results obtained from different metrics is that they are not directly comparable. Disambiguation accuracy is the least strict one, because it treats the tagging task as a much simpler disambiguation task based on a set of gold-standard morphological analyses from the reference corpus. Then, per-

token accuracy and accuracy upper bound are comparable in that they both assume perfect tokenisation. However, contrary to accuracy upper bound, per-token accuracy does not assume correct tagging for incorrectly segmented tokens. Therefore, per-token accuracy is the stricter statistic figure. And as far as the juxtaposition of per-token accuracy and accuracy lower bound is concerned, the latter is the stricter one in theory, because both tokenisation and tagging errors are being penalised. However, taking into account tagger evaluation practice (e.g. (Radziszewski, 2013), accuracy lower bound – while in theory stricter – is being applied to a simpler task than the in the vast majority of studies in non-Polish contexts: taggers evaluated with accuracy lower bound take into account gold-standard morphological analyses contained in the NCP reference corpus (Radziszewski, 2013), while such information is missing in many other corpora, including UDP and thus our own work. As a matter of fact, differences between accuracy lower and upper bound are very small, around 0.3% for all systems evaluated on these metrics (Radziszewski and Acedański, 2002; Radziewski, 2013).

Since in the IXA pipes segmentation/tokenisation and tagging are performed in a sequential manner by two separate modules (the output of the former is the input of the latter), we accordingly decided to base our evaluation not on accuracy upper/lower bound – which would have made our study directly comparable to other Polish taggers – but on two separate evaluations. In doing so, we adhere to the mainstream practice in NLP. The performance of *ixa-pipe-tok*[15] was measured as the overlap of correctly segmented and tokenised sentences with the reference corpus (see 4.1). Tagging accuracy, on the other hand, is measured in terms of overall, unknown and known word accuracy. Given that the OpenNLP library and the IXA pipes compute overall per-token accuracies only, we added the proposed metrics for known/unknown word accuracy in the branch *pos-pl* our fork of the IXA pipes[16].

Summing up, our evaluation scheme treating segmentation/tokenisation and tagging separately is in line with the majority of studies we have reviewed, except for those within the Polish NLP community, where previously disambiguation accuracy and now accuracy upper/lower bound have been used. Consequently, the results of our system are comparable to other Polish taggers only to a limited extent, yet our metric is stricter than accuracy upper bound and in theory less strict than accuracy lower bound, but in the end the latter two yield very similar scores. In exchange, comparability of our results to taggers for languages other than Polish is increased in our approach, which we consider to be very important in view of our ultimate goal to adapt IXA pipes for use with a number of (morphologically complex) languages.

[15] https://github.com/ixa-ehu/ixa-pipe-tok
[16] https://github.com/mustaszewski/ixa-pipe-ml/tree/pos-pl

4 Results and Discussion

4.1 Evaluation of Segmentation and Tokenisation

As the default distribution of *ixa-pipe-tok* has no evaluation module implemented, we added this component to the branch *dev-pl* of our fork[17]. We also added support for Polish by adding non-breaking exceptions for abbreviations. This was done by manually compiling a list of frequent abbreviations that, when followed by a period, do *not* indicate sentence boundaries. The list of almost 700 entries is partially based on our knowledge about Polish and, to a larger extent, on mining NCP and the Polish Wikipedia dump by means of hand-crafted regular expressions using the *grep* command-line utility. The list is, by no means, exhaustive, yet we believe that it sufficiently covers the most frequent Polish abbreviations. In terms of length, the list is similar to those used in *ixa-pipe-tok* for the remaining languages.

Upon evaluation, we obtained an F1 score of 11.7 (precision: 11.9, recall: 11.6) for segmentation/tokenisation accuracy. We believe that this admittedly low score is due the following reasons: Firstly, the current version of *ixa-pipe-tok* fails to detect sentence boundaries when segments lack final punctuation marks but are followed by a blank line, which is a quite frequent case in our data, especially in segments representing headlines, titles or headings. Secondly, a considerable portion of NCP data, namely 7% (Przepiórkowski et al., 2012, p. 33), consists of web content, which is often characterised by non-standard punctuation and capitalisation, as well as tokens representing emoticons or other symbolic content. These non-standard yet frequent special cases pose difficulties to the segmenter/tokenizer, which are beyond the scope of the present thesis project. Therefore, we leave the adaptation of *ixa-pipe-tok* for Polish out for further work.

4.2 Evaluation of Baseline Feature Sets

We have run several hundred tests to determine which combination of local features, i.e. without using cluster features, provides the strongest baseline, both for Maximum Entropy and Perceptron models (for a detailed description of our baseline feature sets, see 3.2.2). As shown in Table 10, the strongest baseline for Maximum Entropy models trained on UDP scores 81.6% overall per-token accuracy on the respective development sets, while for Perceptron models the score is 81.1%. Overall accuracy for the test sets drops slightly. As was expected, accuracy scores for models trained on the NCP are higher than for those trained on UDP, because the former is much larger than the latter, thus providing more training instances.

Another important observation is, not surprisingly, that tagging accuracy for unknown tokens, i.e. tokens that have not appeared in the training data, is much lower than for known tokens. Especially in the case of morphologically complex languages this is a key limitation, because due to the large number of possible morphological word forms per lemma there will inevitably always appear unknown words in test time, notwithstanding the size of the training corpus. A comparison of the performance of Maximum Entropy vs. Perceptron models shows that overall scores are quite similar. For models trained on NCP, unknown word

[17] https://github.com/mustaszewski/ixa-pipe-tok/tree/dev-pl

accuracy is much higher for Perceptron models (approx. +6%), while differences in overall and known word accuracy are approximately 1% only.

Test Set	Per-token accuracy					
	Maximum Entropy			Perceptron		
	Total	Known	Unknown	Total	Known	Unknown
UDP Dev	81.6	88.6	57.9	81.1	87.7	58.6
UDP Test	80.6	88.0	56.0	80.9	87.4	59.2
NCP Dev	87.2	89.8	59.0	88.5	90.6	65.0
NCP Test	86.6	89.3	58.6	87.9	90.2	64.6
P60 Essays (UDP)	71.9	81.5	50.4	72.4	81.4	52.5
P60 Fiction (UDP)	77.6	85.4	51.8	78.7	85.2	56.9
P60 News (UDP)	75.2	84.6	53.2	73.7	82.9	52.3
P60 Plays (UDP)	80.2	86.4	49.6	80.3	85.6	54.0
P60 Science (UDP)	75.4	82.6	55.7	75.2	82.0	56.8
P60 Essays (NCP)	83.5	86.6	59.3	85.0	87.6	65.3
P60 Fiction (NCP)	86.5	89.4	59.1	87.8	90.2	65.2
P60 News (NCP)	85.4	88.1	61.2	86.5	88.7	66.4
P60 Plays (NCP)	88.3	90.3	59.1	89.3	90.8	66.6
P60 Science (NCP)	85.9	87.9	61.7	87.1	88.5	68.8

Table 10: Evaluation of our baselines, including out-of-domain evaluation on five subsets of P60.

Out-of-domain evaluations on the five subsets of P60 show that except for P60 Plays accuracy drops strongly, between -3% and -8.7% on UDP models using Maximum Entropy and between -2.2% and -8.5% for Perceptron models, The decline for models trained on NCP is less pronounced (between -0.1% and -3.1% for Maximum Entropy and between -0.1% and -2.9% for Perceptron models); in the case of P60 Plays accuracy even *improves* slightly. It is noteworthy that unknown word accuracy increases across all five subsets of P60 for NCP models as compared tests on NCP Dev/Test, up to +3.1% for Maximum Entropy models and +4.2% for Perceptron models. We believe that apart from sheer corpus size the reason for the more robust out-of-domain performance of NCP-trained models as compared to UDP-trained models can be attributed to the composition of NCP. Thus, NCP is a balanced corpus containing data from a large variety of subject domains and text types, while UDP is a subset of NCP. According to its documentation, the Polish dependency treebank – upon which UDP is based – was annotated on a randomly selected 20,000 sentences sample of NCP (Woliński, Głowińska, and Świdziński, 2011), which may suggest that certain text types and domains are not as representatively contained in UDP as they are in NCP. Taken together, these observations seem to confirm the high quality of the NCP.

Interestingly, the decline for Maximum Entropy models when tested on P60 (NCP) is slightly stronger than for Perceptron models, which may be linked to the better performance of Perceptron models in terms of unknown word accuracy.

All in all, our baselines are way below the current state of the art. From a practical perspective this means that with such low tagging scores, the baseline itself is not useful for real-world applications at all. At the same time, from the perspective of our research design we believe that the obtained baselines are relatively strong, because we conducted a large number of experiments on parameter tuning to find out how high tagging accuracy scores can get by using local features only. In other

words, the baselines are most likely very close to the upper bound of tagging accuracy that can be achieved by the IXA pipes for a morphologically complex language such as Polish in an out-of-the-box fashion without substantial system modifications. Our approach of initial feature fine-tuning definitely bears the risk of overfitting the baselines to our data. While this would be, without doubt, a problem if the goal of our work was to develop a readily usable tagger, it is less problematic in our research setting that aims to shed light on the potential and limitations of clustering features for morphosyntactic tagging in a semi-supervised system. To counterbalance the disadvantages of parameter tuning, we conducted out-of-domain tests to investigate whether the approach is robust across datasets. We have seen that out-of-domain performance does drop, although not dramatically, especially in the case of models trained on NCP, which appear to be robust across data-sets in terms of unknown word accuracy.

4.3 Evaluation of the Effect of Clustering Features

In order to determine how each of the three word cluster types (Brown, Clark and word2vec) contributes to the tagging task, we added clustering features on top of the two strongest baselines, whose results are presented in the previous section. As for each cluster type and each corpus a number of cluster lexica with different amounts of cluster classes have been induced (42 cluster lexica in total, see section 3.4), for each cluster type we ran a series of tests. The tests consisted in adding the available cluster lexica on top of the baseline feature set: this was done for each of the cluster lexica individually, as well as for combinations of lexica. In all tests, models were trained on the UDP training set and evaluated on the UDP development set, while no tests using NCP data were performed. The reason for this is purely practically motivated: due to the large number of tests, we opted for the much faster training-testing cycles using smaller datasets.

For the sake of clarity, in the following we report results for each of the three cluster types as well as their combinations separately.

4.3.1 Brown Clusters

Brown clusters of 500, 1000 and 2000 classes for each of the four corpora (Araneum, Wiki, Sejm, Pelcra) have been trained, thus resulting in a total of 12 cluster lexica. We ran one test for each of it added atop of the two baselines, plus six tests experimenting on lexica combinations.

As can be seen in Table 11, overall word accuracy may improve the baseline thanks to Brown clusters by 2.2% for Maximum Entropy and by 4% for Perceptron models. Known word accuracy scores improved by up to 2.3% (Maximum Entropy) and 2.8% (Perceptron). Most notably, unknown word accuracy increased substantially: +3.9% for Maximum Entropy and 8.6% for Perceptron models. Taking into consideration that in most cases that yielded the biggest improvements on unknown word accuracy also the highest overall word accuracy values have been obtained, it appears that better performance on out-of-vocabulary words contributes stronger to overall tagging accuracy than improvements on known words.

Cluster(s)	Maximum Entropy			Perceptron		
	Total	**Known**	**Unkn.**	**Total**	**Known**	**Unkn.**
Baseline	81.6	88.6	57.9	81.1	87.7	58.6
Araneum 500	83.9	*90.8*	60.4	84.7	***90.5***	65.3
Araneum 1000	***84.0***	90.5	***61.8***	***85.1***	90.4	***67.2***
Araneum 2000	83.5	90.3	60.2	85.0	90.3	67.1
Araneum 500, 1000, 2000	80.7	88.4	54.8	82.7	88.7	62.5
Wiki 500	*83.8*	90.6	*60.7*	84.1	89.9	64.6
Wiki 1000	83.4	90.2	60.1	*85.1*	*90.4*	*67.1*
Wiki 2000	83.7	***90.9***	59.1	84.7	90.2	66.3
Wiki 500, 1000, 2000	80.8	89.0	52.8	81.9	88.5	59.5
Sejm 500	*82.8*	*90.3*	*57.4*	84.0	*89.8*	*64.1*
Sejm 1000	82.3	89.9	56.4	83.5	89.3	64.1
Sejm 2000	82.5	90.2	56.4	*84.0*	89.4	65.5
Sejm 500, 1000, 2000	78.8	87.6	48.9	81.5	87.8	60.4
Pelcra 500	81.5	89.5	54.1	82.3	88.9	59.8
Pelcra 1000	81.6	*89.8*	53.7	*82.9*	*89.2*	*61.2*
Pelcra 2000	80.9	89.4	52.0	82.4	88.8	60.8
Pelcra 500, 1000, 2000	77.7	87.3	45.1	80.2	87.3	56.0
Araneum 500, Pelcra 500, Sejm 500, Wiki 500	79.7	87.4	53.8	81.1	87.3	59.9
Araneum 1000, Pelcra 1000, Sejm 1000, Wiki 1000	80.0	87.7	53.9	81.5	87.8	60.1
Araneum 2000, Pelcra 2000, Sejm 2000, Wiki 2000	79.8	88.1	51.4	81.3	88.0	58.6
Araneum 1000, Wiki 1000	*82.3*	*89.6*	57.6	*83.8*	*89.3*	*65.2*
Araneum 500, Pelcra 1000	79.7	88.6	49.6	81.3	88.0	58.3
Araneum 500, Wiki 500, Pelcra 1000, Sejm 1000	78.2	87.1	47.9	83.4	89.2	63.9

Table 11: Measuring the effect of Brown clusters added on top of both baselines. Models trained on UDP Train and evaluated on UDP Dev. Scores are reported for overall, known and unknown (= unkn.) word accuracy in percent. Best scores for each corpus are indicated in italics if the scores improve in comparison to the baseline, best scores across all corpora are highlighted in bold italics.

The same cluster lexicon, Brown Araneum 1000, yielded best scores for both Maximum Entropy and Perceptron models. A closer look on which Brown clusters contribute most suggests that the improvement depends on the size of the corpus for unlabelled cluster induction: Araneum and Wiki are the two biggest corpora in our experiments. On the other hand, the smallest corpus, Pelcra, worsened scores for Maximum Entropy models and only slightly improved Perceptron models. A further important observation is that cluster feature stacking (Agerri and Rigau, 2016) does not improve results if only one type of clusters is being used. This appears to be true of stacking cluster lexica of various class numbers obtained from the same corpus, as well as of stacking cluster lexica across corpora. This is relevant insofar as combining clusters increases model complexity and thus training time, computational cost as well as model size.

4.3.2 Clark Clusters

A limiting factor to the experiments on the effect of Clark clusters was extensive cluster induction time (see 3.4). Therefore, we induced cluster lexica only for the Araneum corpus (100 and 200 classes) and the Sejm corpus (100, 200, 400 and 600 classes). As shown in Table 12, the combination of Araneum 100 and 200 increases all three accuracy metrics most. The improvement of total word accuracy

amounts to +3.7% for Maximum Entropy and +4.5% for Perceptron models, which is an even stronger effect than for Brown clusters. Most importantly, unknown word accuracy increases by up to 10%.

Contrary to Brown clusters, the combination of clusters does seem to have an added value for tagging accuracy, both with regard to within-corpus and across-corpus combinations of cluster lexica. What can also be seen is that the much smaller Sejm corpus contributes less to unknown word accuracy and consequently less to overall word accuracy.

	Maximum Entropy			**Perceptron**		
	Total	**Known**	**Unknown**	**Total**	**Known**	**Unknown**
Baseline	81.6	88.6	57.9	81.1	87.7	58.6
Araneum 100	84.3	90.5	63.6	84.8	90.1	66.7
Araneum 200	84.4	90.2	64.5	85.1	90.3	67.5
Araneum 100, 200	***85.3***	***90.7***	***66.8***	***85.6***	***90.7***	***68.6***
Sejm 100	83.3	89.8	60.9	83.3	89.4	62.5
Sejm 200	83.5	89.8	62.0	83.7	89.6	63.4
Sejm 400	83.2	89.6	61.4	83.7	89.5	64.1
Sejm 600	83.5	89.9	62.0	83.3	89.1	63.5
Sejm 100, 200	83.8	90.1	62.1	83.9	89.9	63.6
Sejm 400, 600	83.8	90.0	62.7	84.0	89.6	64.8
Sejm 100, 200, 400, 600	83.9	90.3	62.2	*84.3*	*90.0*	*65.2*
Sejm 100, Sejm 600	*84.0*	*90.2*	*63.1*	83.7	89.5	64.3
Araneum 100, Sejm 100	84.6	90.5	64.6	85.0	90.7	65.9
Araneum 100, Sejm 200	84.7	90.6	64.5	85.1	90.3	67.5
Araneum 200, Sejm 200	*85.1*	*90.7*	*65.9*	85.0	90.1	67.6
Araneum 200, Sejm 100	84.7	90.5	65.0	85.2	90.5	67.0
Araneum 100, Sejm 600	84.8	90.7	64.5	*85.6*	*90.7*	*68.3*

Table 12: Measuring the effect of Clark clusters added on top of both baselines. Models trained on UDP Train and evaluated on UDP Dev. Scores are reported for overall, known and unknown word accuracy in percent. Best scores for each corpus are indicated in italics if the scores improve in comparison to the baseline, best scores across all corpora are highlighted in bold italics.

4.3.3 word2vec Clusters

Since the induction of word2vec clusters is very fast, we had at our disposal a large set of word2vec clusters. However, our results shown in Table 13 suggest that compared to Brown and Clark clusters this cluster type is not as useful for tagging, since the highest improvement on word accuracy accounts for only +1.4% for Maximum Entropy and +1.7% for Perceptron, respectively. In the same way, unknown word accuracy improved only by +2.2% for Maximum Entropy and 4.4% for Perceptron models, which is less than in the case of Brown and Clark clusters. All in all, this cluster type appears to work better with Perceptron models, especially in the case of unknown word accuracy. Similar to the other cluster types, its influence appears to depend on the size of the unlabelled corpus for cluster induction.

Cluster(s)	Maximum Entropy			Perceptron		
	Total	**Known**	**Unknown**	**Total**	**Known**	**Unknown**
Baseline	81.6	88.6	57.9	81.1	87.7	58.6
Araneum 100	82.3	89.0	59.7	82.2	88.3	61.6
Araneum 200	82.6	89.2	60.1	82.3	88.6	60.8
Araneum 400	82.4	89.4	58.6	82.1	88.3	61.0
Araneum 800	82.6	89.5	59.2	82.1	88.1	61.4
Araneum 1000	***83.0***	89.7	***60.1***	82.4	88.3	62.3
Araneum 100, 200, 400	82.1	89.2	58.3	82.5	88.8	61.5
Araneum 800, 1000	82.9	***89.9***	59.3	82.6	88.5	62.6
Araneum 100, 200, 400, 800, 1000	82.2	89.6	57.1	***82.8***	*88.6*	***63.0***
Wiki 100	81.9	88.8	58.2	81.9	88.2	60.8
Wiki 200	82.1	89.0	58.8	81.4	87.6	60.3
Wiki 400	81.7	88.8	57.6	82.3	88.2	62.6
Wiki 800	82.1	89.2	57.9	82.0	88.3	60.6
Wiki 1000	82.4	89.3	*59.0*	82.4	*88.5*	61.5
Wiki 100, 200, 400	81.7	89.0	57.2	*82.4*	88.2	*62.7*
Wiki 800, 1000	*82.5*	*89.6*	58.6	82.1	88.4	60.4
Wiki 100, 200, 400, 800, 1000	81.5	89.2	55.3	81.9	88.1	60.9
Sejm 100	*82.2*	88.9	*59.7*	81.8	88.0	60.8
Sejm 200	82.2	89.0	58.9	82.0	88.2	60.9
Sejm 400	82.1	89.2	57.9	81.7	88.1	60.2
Sejm 800	81.9	89.0	57.9	81.7	88.2	59.6
Sejm 1000	82.2	89.2	58.5	81.6	88.0	60.1
Sejm 100, 200, 400	81.7	*89.3*	56.1	81.9	87.7	*62.3*
Sejm 800, 1000	81.8	89.1	56.9	82.1	88.2	61.4
Sejm 100, 200, 400, 800, 1000	81.0	89.2	53.0	*82.3*	***88.6***	60.6
Pelcra 100	81.9	88.8	58.6	81.3	87.9	58.9
Pelcra 200	*82.1*	88.9	*58.9*	81.5	88.1	59.2
Pelcra 400	82.0	88.9	58.8	81.3	87.9	59.0
Pelcra 800	81.9	88.9	58.1	81.3	*88.2*	58.0
Pelcra 1000	81.7	88.9	57.5	81.3	87.7	59.3
Pelcra 100, 200, 400	81.6	89.4	55.1	*81.7*	87.9	*60.6*
Pelcra 800, 1000	81.2	88.7	56.0	81.5	87.9	59.7
Pelcra 100, 200, 400, 800, 1000	80.7	*89.1*	52.2	81.3	87.9	58.8
Araneum 200, Wiki 200, Pelcra 200	*82.7*	*89.4*	*59.7*	81.9	88.2	60.8
Pelcra 400, Sejm 400	82.1	89.4	57.5	82.1	88.6	60.0
Araneum 100, Wiki 100	82.0	88.9	58.5	*82.8*	*88.6*	*62.9*
Sejm 100, Pelcra 100	82.2	89.1	58.6	82.3	88.3	62.2
Araneum 100, Pelcra 100	82.3	89.1	59.2	82.4	88.6	61.3

Table 13: Measuring the effect of word2vec clusters added on top of both baselines. Models trained on UDP Train and evaluated on UDP Dev. Scores are reported for overall, known and unknown (= unkn.) word accuracy in percent. Best scores for each corpus are indicated in italics if the scores improve in comparison to the baseline, best scores across all corpora are highlighted in bold italics.

In conclusion, the effect of word2vec clusters on morphosyntactic tagging is much lower than in the case of the other two cluster types. However, their usefulness especially for unknown word accuracy is not to be dismissed, because they can be induced in very short time and the resulting cluster lexica are the smallest ones in terms of size.

4.3.4 Combining Cluster Types

After determining how each cluster type contributes to tagging as reported in the previous three subsections, we stacked and combined the most promising cluster lexica from each cluster type. The results of these tests are shown in Table 14. From this, it becomes obvious that word2vec clusters have a minor effect on the overall task, since the highest scores obtained from combinations that leave out word2vec clusters are very similar to those that do employ word2vec clusters. It can also be seen that the combination of Brown and Clark clusters performs best, with an overall word accuracy score of 86.2%, which equals to an improvement by 5.1% compared to the baseline for Perceptron models. In contrast, the best score for the use of one cluster type (85.6%) was obtained by combining Clark Araneum 100 and Clark Araneum 200 clusters (see 4.3.2). Furthermore, stacking clusters increased unknown word accuracy scores by +11.2%. Taken together, this suggests that there is, indeed, an added effect of cluster stacking, as was confirmed by Agerri and Rigau (2016) for NERC.

Cluster Combinations			**MaxEnt**			**Perceptron**		
			T	**U**	**K**	**T**	**U**	**K**
Baseline			81.6	88.6	57.9	81.1	87.7	58.6
Brown	**Clark**	**w2v**						
Aran 1000	Aran 200	Aran 1000	84.5	90.9	62.9	85.6	90.6	68.7
Wiki 1000	Aran 200	Aran 1000	84.3	90.8	62.5	86.1	90.9	69.8
Sejm 500	Aran 200	Aran 1000	84.4	91.0	61.8	85.3	90.5	67.5
Aran 1000, Wiki 500	Aran 100, 200	Wiki 1000	83.3	90.1	60.2	84.6	89.7	67.3
Aran 1000	Aran 100, 200	Aran 1000	***84.8***	*91.0*	***63.7***	***86.1***	*91.0*	***69.8***
Aran 1000	Aran 100, 200	--	84.6	91.0	62.8	*86.1*	***91.1***	*69.5*
Aran 500	Aran 100, 200	--	*84.7*	*91.1*	*62.9*	85.6	90.8	68.0
Aran 1000, Wiki 500	Aran 100, 200	--	83.0	89.9	59.7	84.4	89.6	66.8
Wiki 500	Aran 100, 200	--	84.5	90.8	63.0	*86.0*	*90.9*	69.6
Wiki 1000	Aran 200	--	84.4	90.9	62.2	85.6	90.8	67.8
Wiki 1000	Aran 100, 200	--	*84.7*	***91.1***	*63.0*	85.9	90.6	*69.8*
Sejm 500	Aran 200	--	84.0	90.9	60.8	85.2	90.4	67.6
Sejm 500	Aran 100, 200	--	*84.4*	*91.1*	*61.8*	*85.4*	*90.6*	*67.7*
Sejm 1000	Aran 100, 200	--	84.0	90.8	61.0	85.2	90.3	64.2

Table 14: Measuring the effect of combining Brown, Clark and word2vec clusters added on top of baselines. Models trained on UDP Train and evaluated on UDP Dev. Scores for overall (T), known (K) and unknown (U) word accuracy in %. Best scores for each combination group are indicated in italics if they improve the baseline, best scores across all corpora are highlighted in bold italics.

4.4 Final Evaluation

The final step of our experiments consisted in an evaluation of the two best-performing combinations of clustering features. For Maximum Entropy training, Clark clusters of 100 and 200 classes obtained from the Araneum corpus were chosen for the final evaluation, whereas for Perceptron training we combined Brown clusters of 1000 classes, Clark clusters of 100 and 200 classes, and word2vec clusters of 1000 classes, all of which were, again, obtained from the Araneum.

4.4.1 Evaluation on Test Sets

Models trained on UDP Train were evaluated on UDP Test (see Figure 2), whereas those trained on NCP Train were evaluated on NCP Test (see Figure 3). As already mentioned in section 4.2, tagging accuracy of the baselines slightly drops upon evaluation on the test sets in comparison to the corresponding development sets.

As can be seen in the two bar charts, the clustering features consistently improve the baseline for all three metrics (total, known and unknown word accuracy), with the highest improvement being +5.1% on total and +10.3% on unknown word accuracy for Perceptron models trained on UDP. In absolute figures, our best scores for overall word accuracy are 86.0% on UDP Test and 90.0% on NCP Test. In general, our system appears to work slightly better with Perceptron than with Maximum Entropy models.

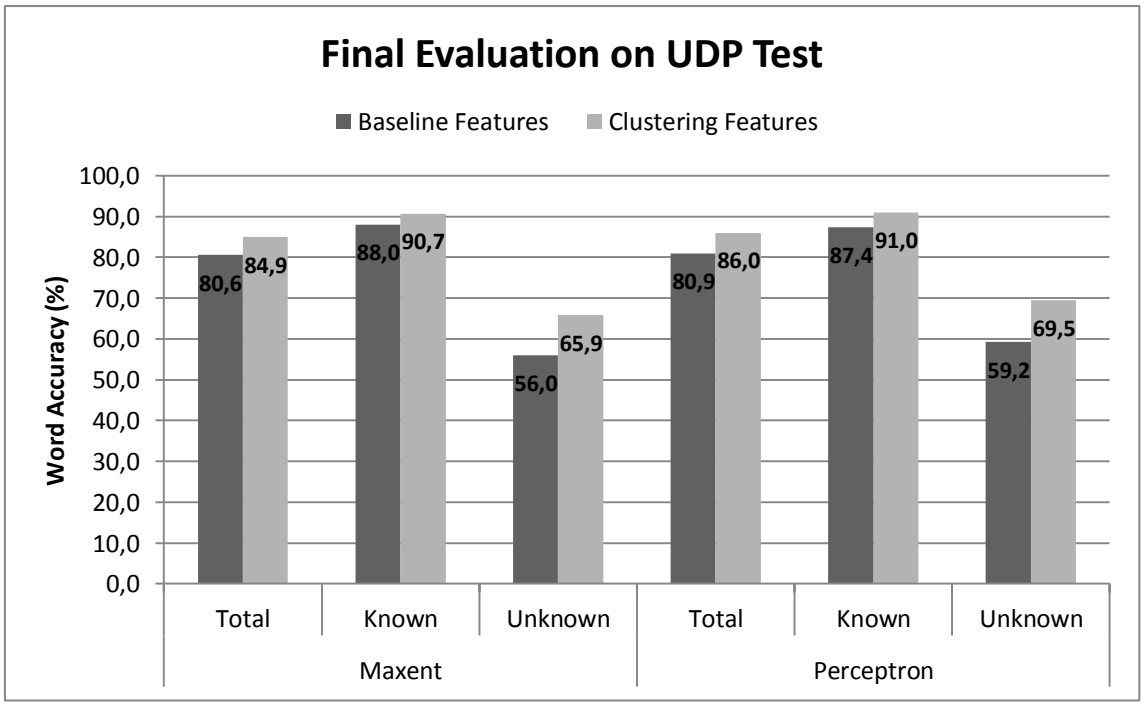

Figure 2: Results for the two best-performing combinations of local and clustering features. Models were trained on UDP Train and evaluated on UDP Test.

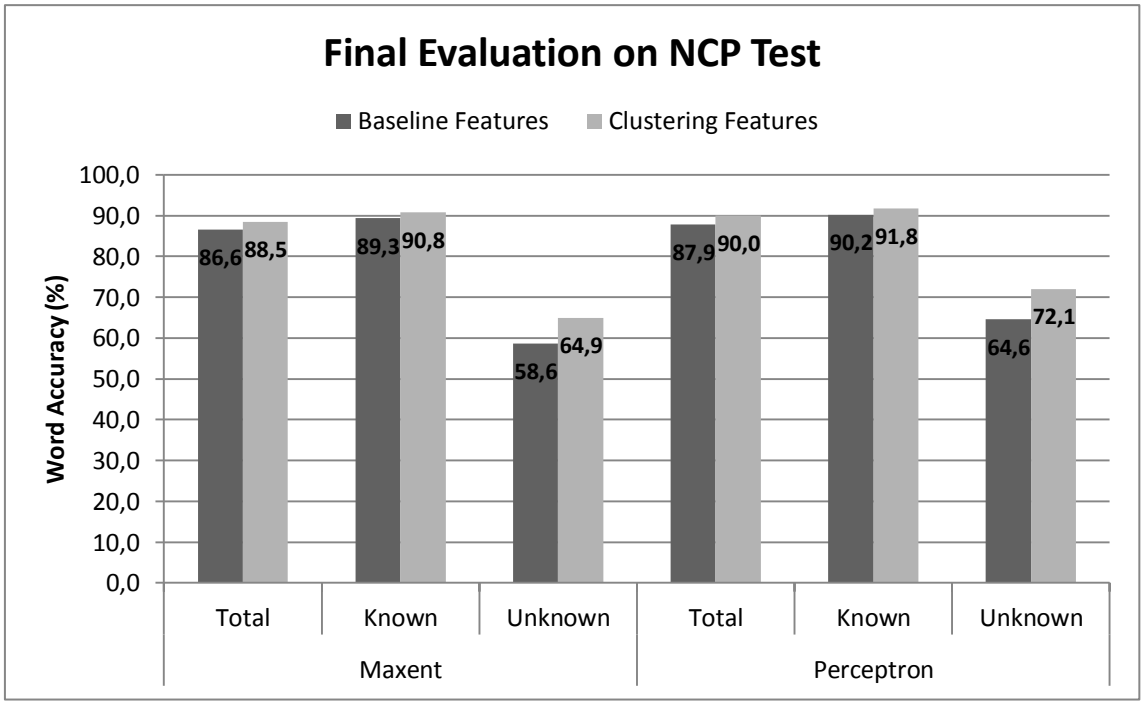

Figure 3: Results for the two best-performing combinations of local and clustering features. Models were trained on NCP Train and evaluated on NCP Test.

Apart from accuracy scores, further important parameters of system performance are not to be left out in our evaluation: training time and size of the models, which are shown in Table 15. Especially in the case of Perceptron models, which rely on a combination of 4 cluster lexica, training time and size is rather large when compared to the baselines using local features only. Another factor to be taken into consideration are memory issues. In our system, Perceptron models require a lot of working memory in training, for the best-scoring model trained on NCP we needed 120 GB for indexing the training data set, which can be explained by the large size of the training corpus and the set of outcome tags. A similar problem was reported by Radziewski (2013) for CRF models, too. Maximum Entropy training, on the other hand, is less problematic in terms of memory use; apart from that, it supports multithreading.

	Local Features Only				**Local + Clustering Features**			
	Maximum Entropy		Perceptron		Maximum Entropy		Perceptron	
	UDP	NCP	UDP	NCP	UDP	NCP	UDP	NCP
Time (hh:mm)	00:21	01:47	00:10	06:15	00:12	01:59	00:14	17:54
Size (MB)	3	15	4	30	86	98	109	135

Table 15: Training time and size of best-performing models.

4.4.2 Out-of-Domain Evaluation

Finally, we performed out-of-domain evaluations. In the out-of-domain evaluation of our baselines, we identified a drop in tagging accuracy as compared to the development sets (see 4.2) The general picture for our models using combinations of local and clustering features is very similar: accuracy drops, too, although the decrease is less pronounced (see Table 16 and Table 17). We also observed that the Plays subset of P60 was least prone to accuracy declines, just as in the case of our

baselines. What is more, the positive effect of clustering features on tagging of unknown words was observed in out-of-domain evaluation, too. The magnitude of improvement on unknown vocabulary in out-of-domain evaluation is very similar to the one observed when determining the effect of each cluster type on the overall task, where the improvement on unknown vocabulary as compared to the use of local features only was as high as 10% and more for some configurations. Also, the robustness of unknown word accuracy for NCP models on all five subsets of P60 was observed, just as was the case with the baselines.

		Maximum Entropy			**Perceptron**		
		Total	**Known**	**Unknown**	**Total**	**Known**	**Unknown**
UD Test	Baseline	80.6	88.0	56.0	80.9	87.4	59.2
	Clusters	84.9	90.7	65.9	86.0	91.0	69.5
P60 Essays	Baseline	71.9	81.5	50.4	72.4	81.4	52.5
	Clusters	77.8	84.9	61.9	78.3	85.2	63.1
P60 Fiction	Baseline	77.6	85.4	51.8	78.7	85.2	56.9
	Clusters	81.7	87.9	61.1	83.1	88.4	65.6
P60 News	Baseline	75.2	84.6	53.2	73.7	82.9	52.3
	Clusters	79.6	87.4	61.4	79.9	87.1	63.2
P60 Plays	Baseline	80.2	86.4	49.6	80.3	85.6	54.0
	Clusters	82.8	87.9	57.8	83.2	87.5	62.4
P60 Scientific	Baseline	75.4	82.6	55.7	75.2	82.0	56.8
	Clusters	79.6	85.7	63.0	80.5	85.8	66.1

Table 16: Out-of-domain evaluation for models trained on UDP Train using both local and clustering features. Scores are presented in total, known and unknown word accuracy in percent.

		Maximum Entropy			**Perceptron**		
		Total	**Known**	**Unknown**	**Known**	**Total**	**Unknown**
NCP Test	Baseline	86.6	89.3	58.6	87.9	90.2	64.6
	Clusters	88.5	90.8	64.9	90.0	91.8	72.1
P60 Essays	Baseline	83.5	86.6	59.3	85.0	87.6	65.3
	Clusters	86.0	88.6	66.3	87.5	89.4	72.7
P60 Fiction	Baseline	86.5	89.4	59.1	87.8	90.2	65.2
	Clusters	88.6	91.2	63.9	90.0	91.9	72.0
P60 News	Baseline	85.4	88.1	61.2	86.5	88.7	66.4
	Clusters	87.5	89.8	66.7	88.7	90.5	72.3
P60 Plays	Baseline	88.3	90.3	59.1	89.3	90.8	66.6
	Clusters	89.8	91.5	64.5	91.0	92.2	73.2
P60 Scientific	Baseline	85.9	87.9	61.7	87.1	88.5	68.8
	Clusters	87.7	89.4	67.0	88.9	90.1	73.4

Table 17: Out-of-domain evaluation for models trained on NCP Train. Scores are presented in total, known and unknown word accuracy in percent.

4.4.3 Interpretation and Discussion of Results

The two major conclusions to be drawn from our data are the following: First, the strongest effect of clustering features is on unknown word accuracy, which can be easily seen in the bar charts (Figure 2 and Figure 3). This is of great relevance for highly inflected languages, where data sparseness is mainly caused by the large number of word forms per lemma, many of which will inevitably be unknown to models. While clustering features alone are not enough to push a baseline signifi-

cantly beyond state-of-the-art tagging results, our results suggest that they are a simple yet very efficient technique that, when added to a system, may help to better deal with morphologically caused data sparsity and to improve a system's robustness across datasets outside of the training domain. Eventually, this may help to improve the current state of the art if used in conjunction with other techniques reported in the literature.

The second major conclusion is that the effect of clustering features is stronger for models trained on smaller datasets. The bar charts clearly show that the improvement of clustering features relative to the respective baselines is consistently higher for models trained on the much smaller UDP corpus (Figure 2) than for those trained on NCP (Figure 3). What is more, cluster features for Perceptron models trained on UDP helped to improve overall tagging accuracy from 80.9% to 86.0%, which is close to the baseline of 87.9% for NCP-trained models. Taking into consideration that UDP Train is only approximately 7% the size of NCP Train these figures are quite impressive. For unknown word accuracy, cluster features push scores for UDP models even from 59.2% to 69.5%, which is almost 5% beyond the baseline for NCP models of 64.6%. All this suggests that clustering features have the potential to alleviate the dependency on large hand-annotated training corpora, therefore corroborating in the context of morphosyntactic tagging what Agerri and Rigau (2016) found for NERC. That said, we do not imply that clustering features are not useful in situations where large training corpora are available. On the contrary, we have observed positive effects on NCP data, too, albeit not as strong ones as in the case of the smaller UDP. Consequently, we conclude that the stronger the baseline, the less pronounced the effect of clustering features on overall word accuracy. Still, clusters do have an impact on unknown word accuracy, even if models were trained on large, balanced and high-quality corpora.

Summing up, we would like to outline two scenarios, in which in our opinion the application of clustering features should be considered for morphological tagging. Firstly, in the absence of large training corpora, clustering may alleviate dependency on them, thus improving both overall and unknown accuracy. This may prove useful especially for less-resourced languages that lack such resources, or for specific subject domains or text types that are otherwise underrepresented in training data. Secondly, if large training corpora are available, clustering features may help to improve, first and foremost, tagging accuracy on unknown words. This may be the case when taggers are to be applied to domains other than the training domain (which is, in fact, a frequent application scenario), or when training corpora are not balanced. In this scenario, the main benefit is on out-of-domain robustness. Not to be neglected is the effect on dealing with data sparsity inherent to highly inflected languages, which applies to either of the two outlined scenarios.

4.4.4 Comparison to Other Systems

Based on a literature review, we compare our results to other systems, both for Polish and for other morphologically complex languages. Our goal is not to provide an exhaustive comparison, but an anecdotal one for a general overview. A major limitation to the comparison is the use of different metrics for most Polish taggers (see discussion in 3.5) on the one hand and that papers do not always report all relevant figures on the other.

Systems for Polish

We compared our work the following taggers for Polish: *WCRFT* (Radziszewski, 2013), *Pantera* (Acedański, 2010), *WMBT* as a stand-alone tagger and as part of an ensemble tagger (Radziszewski and Śniatkowski, 2011), and *UDPipe* (Straka et al., 2016). All systems are trained and tested on NCP data, except for *UDPipe*, which uses UDP data.

	Word.Ac		**Dis.Ac**	**Ac.Low.B**	**Ac.Up.B**	**Ac.Low.B.Unk**	**Tokens**	**|Tagset|**
	Total	**Unkn.**						
ours (NCP)	*90.0*	*72.1*	--	--	--	--	*962 K*	*901*
WCRFT	--			90.3	90.7	40.1	1.1 M	--
PANTERA	--	--	92.4	88.8*	89.1*	14.7*	--	--
WMBT	--	--	93.0	87.5*	87.8*	13.6*	1.1 M	--
WMBT ens.	--		94.1	--	--	--	1.1 M	--
ours (UDP)	*86.0*	*69.5*	--	--	--	--	*69 K*	*607*
UDPipe	84.7	--	--	--	--	--	69 K	607

Table 18: Comparison with other systems for Polish. Scores indicated in percent for total and unknown word accuracy (Word.Ac.), disambiguation accuracy (Dis.Ac), accuracy lower bound (Ac.Low.B), accuracy upper bound (Ac.Up.B), accuracy lower bound for unknown words (Ac.Low.B.Unk). Size of training sets indicated in tokens and tagset cardinality in unique tags. Scores marked with * are based on re-evaluation by Radziszewski and Acedański (2012).

The only fully comparable system is *UDPipe* since it was trained/tested on exactly the same data and is evaluated using word accuracy. As shown in Table 18, our system outperforms it by 1.3%. *UDPipe* is a language-independent system that uses exactly the same architecture and settings for each of the 32 supported languages. In contrast, our system entailed some feature tuning for Polish, albeit only to a very limited degree. The main difference between both systems is the use of a morphological guesser in *UDPipe*. This demonstrates how powerful clustering features are, because without any morphological analyser our system outperforms one that uses a guesser. This, in turn, suggests that the extremely simple approach employed in our work has the potential to replace, at least in parts, a more complex morphological analysis.

With regard to the remaining Polish taggers, the picture is somewhat obscured because of the difficulties related to the use of different evaluation metrics. Although the results are not directly comparable, it appears that our system is at least as competitive as current state-of-the-art taggers for Polish, because in practice word accuracy is the strictest of all metrics (see discussion in 3.5). Most strikingly, our system shows huge improvements in terms of tagging performance of unknown words. This is remarkable given that all Polish taggers strongly rely on morphological analysis while our system only applies unsupervised clustering features.

Systems for Other Languages

In Table 19, we give a comparison with systems for other morphologically rich (Slavic) languages that report results in terms of per-token accuracy. However, it must be noted that due to differences in the linguistic characteristics of each language as well as different corpora sizes scores are not directly comparable.

		Word Accuracy		Tokens	|Tagset|
		Total	Unknown		
this work (NCP)	*PL*	*90.0*	*72.1*	*962 K*	*901*
Spoustová et al. (2009)	CZ	95.9	--	1.5M	--
Loftsson & Östling (2013)	IS	93.8	--	590 K	565
Halácsy et al. (2007)	HU	98.4	96.0	1M	--
this work (UDP)	*PL*	*86.0*	*69.5*	*69 K*	*607*
Agić et al. (2013)	HR	87.7	--	88 K	--
Agić et al. (2013)	SR	85.6	--	88 K	--
Agić et al. (2010)	HR	90.8	72.6	107 K	880
Georgiev et al. (2012)	BG	98.0	--	254 K	552
Silfverberg et al. (2014)	CZ	91.0	77.8	(5K sent.)	908
Silfverberg et al. (2014)	FI	88.7	63.6	(5K sent.)	2,141

Table 19: Comparison with systems for other languages. Scores in percent for total and unknow-word accuracy. Size of training sets indicated in tokens and tagset cardinality in unique tags.

Spoustová et al. (2009), who also try to leverage unlabelled data in a semi-supervised system, obtained very high accuracy scores of almost 96%, which is one of the best scores for morphosyntactic tagging in Slavic languages known to us. With the help of the unsupervised component, they improved their baseline by almost 5%. While their approach is similar to ours in terms of combining supervised and unsupervised components, it differs in as far as it uses a morphological analyser that precedes the tagger. The *IceTagger* (Loftsson and Östling, 2013) for Icelandic was trained on data that is roughly 60% the size of NCP in terms of tokens and tagset cardinality. It is a mature system that achieves high tagging accuracies (93.8%), albeit on a tagset that is much smaller than in Polish. Interesting from our perspective is that it also uses Perceptron models as well as word representations from unlabelled text. Again, the key component that probably explains the high scores is a morphological guesser. The taggers for Croatian and Serbian by Agić et al. (2013) employ a rather simple approach and score in an approximately similar range as our system. The ensemble voting system presented by Agić et al. (2010), scores almost 5% better than our system on a larger training corpus. However, ensemble voting is not as straight-forward as our approach. For Bulgarian, Georgiev et al. (2012) score an extraordinarily high overall tagging accuracy of 98.0%, which is at first sight surprising for a language with such a rich morphology[18] since the scores are very similar to POS tagging accuracies for English. Their system, however, heavily relies on a pre-compiled inflectional lexicon, which dramatically reduces the number of unknown words, which is the very problem of morphosyntactic tagging in highly inflected languages. Another tagger with very high accuracy, both for all (98.4%) and unknown (96.0%) words, is *HunPos* (Halácsy et al., 2007) for Hungarian based on Hidden Markov Models. The system features a strong morphological analyser, thereby considerably reducing the number out-of-vocabulary words. Although Hungarian is agglutinative, this tagger architecture is interesting for inflective languages, too, mainly because of outstandingly short training times. Finally, we reviewed Silfverberg et al.'s (2013) CRF-based tagger using sub-label dependencies as applied to Czech and Finnish. Unfortunately, it is difficult to compare it to our system because corpus size is not explicitly

[18] Compared to Polish, Bulgarian has a simpler nominal inflection but more complex verbal inflection. In view of the size of the cardinality of the tagset (approx. 550), we may assume that the morphological complexity is similar to our Polish case, although the Bulgarian training corpus is more than three times the size of UDP.

provided in their paper. Taking into consideration the cardinality of the tagset as well as the results for the remaining languages presented in the paper it appears to be a rather strong system that may outperform our system in direct comparison. Most importantly, it scores higher for unknown words than our system.

In conclusion, the comparison of related work unfortunately does not yield a clear picture. A comparison with other Polish taggers trained on NCP is not fully feasible because of the use of different accuracy metrics. Nevertheless, our system seems to be at least as competitive as current state-of-the-art systems. Most importantly, our system performs much stronger in terms of unknown word accuracy despite not using a morphological analyser or guesser.

5 Conclusions and Future Work

The main contributions of this work are the following:

1. Without using a component for morphological guessing or analysis, we have proposed a morphosyntactic tagger for Polish that achieves similar or even superior tagging accuracy as compared to more complex state-of-the-art systems. Most notably, our system performs much better on unknown word accuracy than any other Polish tagger.
2. Thanks to the use of clustering features based on distributional semantics, our system employs a simple yet robust approach to deal with data sparsity in a semi-supervised manner. This suggests that clustering features may replace or complement more complex morphological analysers or guessers.
3. Our system is designed to meet the requirement of simplicity and user-friendliness while at the same time being publicly available within the open-source IXA pipes.

We set out to investigate the potential of word clustering features obtained in a purely unsupervised fashion in order to improve supervised morphosyntactic tagging for highly inflected languages, thus resulting in a semi-supervised approach. The assumption underlying our work is that word representations such as word clusters help to address data sparsity inherent to morphosyntactic tagging in morphologically complex languages, with the main problem being out-of-vocabulary words unknown to the model. Our ultimate goal is to propose a simple yet efficient technique to improve tagging in language-independent, trainable, user-friendly and robust tools. To this end, we conducted experiments on Polish datasets using the POS tagging module of the IXA pipes, which was shown to yield state-of-the-art results for languages with less complex morphology than Polish. In general, our study is a replication of the work done on the use of clustering features in NERC (Agerri and Rigau, 2016), but in application to morphosyntactic tagging.

Our proposed approach consists in adding word clustering features induced from large amounts of unlabelled text on top of a baseline system that relies exclusively on local features. Our system can therefore be deemed as linguistically uninformed, because no language-specific component is added to our system. Instead, it attempts to leverage recent advances in unsupervised distributional semantics, which is becoming increasingly popular in view of the public availability of large unlabelled corpora for the vast majority of languages and which has therefore had an impact on almost any field of NLP. To test our assumptions, we first established a baseline by implementing a limited number of features useful for morphosyntactic tagging of highly inflected languages in the IXA pipes and by subsequently determining the best-performing parameter setting for Maximum Entropy and Perceptron models. Then, we used freely available tools to induce three types of word clustering lexica (Brown clusters, Clark clusters and word2vec clusters); for each type we used a total of four corpora (Araneum Polonicum Maius, Polish Wikipedia dump, Polish Sejm Corpus, Polish source texts from PELCRA multilingual parallel corpus) and induced cluster lexica distinguishing various numbers of cluster classes. In the next step, we investigated how each cluster type and corpus contributes to the tagging task. Our results suggest that Brown and even more so Clark clusters are very useful to the task, and that the size of corpora used for cluster induction plays an important role, too. Furthermore, we showed that stacking and combining

cluster features may have an added value as compared to the use of single cluster lexica. Finally, we evaluated the best-performing combinations of cluster features added atop of local features on our test sets, which included five datasets for out-of-domain evaluation.

The main findings of our systematic evaluation are the following: Firstly, clustering features do improve, as predicted, tagging accuracy as compared to baselines. The stronger the baseline, the less pronounced the improvement. In the same way, we observed that the magnitude of improvement depends on the amount of training data: the smaller the training corpus, the higher increases in tagging accuracy, which we suggest has important implications for less-resourced languages on the one hand and for subject domains for which large amounts of training data are not available. Thus, we have observed that thanks to clustering features baseline scores for a system trained on a small corpus can be pushed beyond the baseline for a corpus that is 13 times larger. This may alleviate the dependency on supervised training data considerably. Secondly, using neither a morphological guesser/analyser nor a morphological dictionary, our system is at least as competitive as current state-of-the-art systems for Polish. Most notably, our system scores much better (around 30 %) on unknown word accuracy than other taggers trained on NCP, despite not exploiting gold-standard morphological analysis from the reference corpus. Similarly, without the use of a morphological guesser we outperform *UDPipe* in terms of overall word accuracy. We believe this is due to the effect of clustering features, which suggests that this simple approach can replace, at least in parts, more complex techniques for morphological analysis. It should be noted that our system is efficient and easy to install and use due to its user-friendly design, while at the same time being available under a permissive open-source licence.

A limitation of our study is that we conducted our research only on Polish data, while investigating the effect of clustering features in other morphologically complex languages may have been insightful, too. Furthermore, the use of an accuracy measurement that is not compatible with most other Polish tagger evaluations can be regarded as a limitation, too. However, in exchange we increase comparability of our results with studies conducted on other languages, since the mainstream metrics for Polish seem to be a peculiarity of the Polish NLP community. On the other hand, the differentiated measurement of overall, known and unknown word accuracy is a strength of our study, because it sheds light on the main cause for data sparsity in morphologically complex languages, i.e. vocabulary unknown to models because of the high number of possible word forms per lemma. In the same way, our out-of-domain evaluations on five different datasets are important indicators for systems' robustness across text types and subject domains. Here, reporting of unknown word accuracies is, again, of great importance. A further strength is the use of two different training corpora. Although UDP is a subset of NCP, the insights from this dual approach are important to understand the effect of clustering features since those two corpora not only differ significantly in size, but also in composition: NCP is a balanced corpus, while UDP is a random sample of it. Finally, by making available our system's code as well as our data sets, we guarantee the reproducibility of our results.

As far as future work is concerned, we identify two main directions. Firstly, replications of our results in further (morphologically complex) languages are desirable. Since large unlabelled corpora are available for most languages, a number of

different languages and language typologies might be explored. Secondly, towards our ultimate goal of adapting the IXA pipes to morphologically complex languages, two promising avenues deserve our attention: combining our approach with complementary techniques for morphological analysis, or employing tiered tagging. With regard to morphological analysis, using a static lexicon of inflectional forms means relying on language-specific resources, which is not fully consistent with the otherwise language-independent architecture of IXA pipes. Therefore, the implementation of an (universal) morphological guesser appears more appropriate. Given our observation that clustering features can, to a certain extent, replace a morphological guesser, it will be interesting to see how a combination of these components contributes to the tagging task. No less importantly, we believe that work towards the optimisation of the remaining NLP components of the IXA pipes for Polish, i.e. tokenisation/segmentation, lemmatisation, NERC, parsing and chunking, is important, too, because it may help to provide the Polish NLP community with a simple yet powerful, full-fledged pipeline.

To conclude this work, we would like to give a brief statement on research methodology in POS and morphosyntactic tagging: For the sake of system comparability, we believe that it is of utmost importance to harmonise the way tagger evaluation is being communicated to the research community. Our suggestion is to rigorously report the following figures: differentiated per-token accuracies for known and unknown words apart from overall per-token accuracies; percentage of unknown words in development and test sets; sentence accuracy; size of the vocabulary in terms of tokens, types and lemmas for each split of the corpus (training, development and testing); and the cardinality of the tagset in each of the splits. In addition, we encourage to always perform out-of-domain evaluations on various datasets in order to be able to judge tagger performance in real-world settings. We ourselves did not compute sentence accuracy and percentage of unknown vocabulary, though we realised in the course of our work that it would have been very informative to better understand the behaviour of our tagger. We therefore hope that in the near future comparable best-practice guidelines for tagger evaluation will be followed to facilitate the comparability of the limitations and strengths of individual systems across languages, subject domains and text types. This may eventually promote in a community effort further performance improvement in this crucial NLP task.

References

Acedański, Sz. (2010). A morphosyntactic brill tagger for inflectional languages. In: Loftsson, H., Rögnvaldsson, E. and Helgadóttir, S., (eds.): *Advances in Natural Language Processing*. Berlin, Heidelberg: Springer, pp. 3-14.

Agerri, R., Bermudez, J. and Rigau, G. (2014): IXA pipeline: Efficient and Ready to Use Multilingual NLP tools. In: *Proceedings of the 9th Language Resources and Evaluation Conference (LREC2014)*, pp. 3759-3764.

Agerri, R. and Rigau, G. (2016): Robust multilingual Named Entity Recognition with shallow semi-supervised features. In: *Artificial Intelligence* vol. 238, pp. 63-82.

Agić, Ž., Tadić, M. and Dovedan, Z. (2010): Tagger Voting Improves Morphosyntactic Tagging Accuracy on Croatian Texts. In*: Proceedings of 32nd International Conference on Information Technology Interfaces*, pp. 61-66-

Agić, Ž, Ljubešić, N and Markler, D. (2013): Lemmatization and Morphosyntactic Tagging of Croatian and Serbian. In: *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing*, pp. 48-57.

Baldwin, B. (2006): Why do you Hate CRFs? In: *LingPipe Blog*. https://lingpipe-blog.com/2006/11/22/why-do-you-hate-crfs (retrieved 07/2016).

Benko, V. (2014): Aranea: Yet Another Family of (Comparable) Web Corpora. In Sojka, P., Horák, A., Kopeček, I. and Pala, K. (eds.): *Text, Speech and Dialogue. Proceedings of the 17th International Conference (TSD 2014)*. Springer International Publishing Switzerland, pp. 257-264.

Benko, V. (2016): Two Years of Aranea: Increasing Counts and Tuning the Pipeline. In: *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pp. 4245-4248.

Brants, T. (2000): Tnt: A statistical part-of-speech tagger. In: *Proceedings of the 6th Conference on Applied Natural Language Processing*, pp 224-231.

Brown, P.F., Desouza, P.V., Mercer, R.L., Pietra, V.J.D. and Lai, J.C. (1992): Class-Based n-gram Models of Natural Language. In: *Computational Linguistics* vol. 18, pp. 467-479.

Caselli, T., Vossen, P., van Erp, M., Fokkens, A., Ilievski, F, Izquierdo, R. Le, M. Morante, R. and Postma, M. (2015): When it's all piling up: investigating error propagation in an NLP pipeline. In: Izquierdo, R. (ed.): *Proceedings of the Workshop on NLP Applications: Completing the Puzzle*. Aachen.

Ceauşu A. (2006): Maximum Entropy Tiered Tagging. In: *Proceedings of the 11th ESSLLI Student Session*, pp. 173-179.

Ciaramita, M. and Altun, Y. (2006): Broad-Coverage Sense Disambiguation and Information Extraction witha Supersense Sequence Tagger. In: *Proceedings of the 2006 Conference on Empirical Methods in NLP*, pp. 594-602.

Collins, M. (2002): Discriminative training methods for Hidden Markov Models: *Theory and experiments with perceptron algorithms*. In: EMNLP 2002.

Clark, A. (2003): Combining distributional and morphological information for part of speech induction. In: *Proceedings of the 10th Conference on European Chapter of the Association for Computational Linguistics - Volume 1*, pp. 59-66.

Eger, S. Gleim, R. and Mehler (2016): Lemmatization and Morphological Tagging in German and Latin: A comparison and a survey of the state-of-the-art. In: *Proceedings of the 10th International Conference on Language Resources and Evaluation*, 2016.

Georgiev, G., Zhikov, V., Osenova, P, Simov, K. and Nakov, P. (2012): Feature-rich part-of-speech tagging for morphologically complex languages: application to Bulgarian. In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 492-502.

Giesbrecht, E. and Evert, S. (2009): Is Part-of-Speech Tagging a Solved Task? An Evaluation of POS Taggers for the German Web as Corpus. In: *Proceedings of the 5th Web as Corpus Workshop (WAC5), San Sebastián, Spain*, pp. 27-35.

Giménez, J. and Màrquez (2004): SVMTool: A general POS tagger generator based on support vector machines. In: *Proceedings of the 4th International Conference on Language Resources and Evaluation*.

Hajič, J. (2000): Morphological Tagging: Data vs. Dictionaries. In: *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, pp 94-101.

Halácsy, P., Kornai , A. and Oravecy, Cs. (2007): HunPos: an open source trigram tagger. In: *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pp. 209-212

Jassem, K. (2013): PSI-Toolkit – how to turn a linguist into a computational linguist. In: *Proceedings of the 15th International Conference Text, Speech and Dialogue*, Brno, pp. 215-222.

Kobyliński, Ł. (2013): Improving the accuracy of Polish POS tagging by using voting ensembles. In: *Proceedings of the 6th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pp. 453-456.

Kobyliński, Ł. (2014): PoliTa: A multitagger for Polish. In: *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*, pp. 2949-2954.

Liang, P. (2005): *Semi-Supervised Learning for Natural Language*. MA Thesis. Massachusetts Institute of Technology.

Loftsson, H. and Östling, R. (2013): Tagging a Morphologically Complex Language using an Averaged Perceptron Tagger: The Case of Icelandic. In: *Proceedings of the 19th Nordic Conference of Computational Linguistics*, pp. 105-119.

Manning, C.D. (2011): Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics? In: Gelbukh, A. (ed.): *Computational Linguistics and Intelligent Text Processing, 12th International Conference, CICLing 2011, Proceedings, Part I*. Springer, pp. 171-189.

Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J. and McClosky. D. (2014): The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60.

Marcus, M.P., Santorini, B. and Marcinkiewicz, M.A. (1993): Building a large annotated corpus of English: The Penn treebank. In: *Computational Linguistics* vol. 19, pp. 313-330.

McCallum, A (2002): *MALLET: A Machine Learning for Language Toolkit*. http://mallet.cs.umass.edu.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J. (2013): Distributed representations of words and phrases and their compositionality. In: *Advances in Neural Information Processing Systems,* pp. 3111-3119.

Müller, T., Schmid, H. and Schütze, H. (2013). Efficient higher-order CRFs for morphological tagging. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 322-332.

Nivre, J. et al. (2016): Universal Dependencies v1: A Multilingual Treebank Collection. In: *Proceedings of the 10th Language Resources and Evaluation Conference (LREC 2016)*, pp. 1659-1666.

Ogrodniczuk, M. (2003): Wzbogacenie korpusu słownika frekwencyjnego o nowe kody gramatyczne. In: Bień, J.S. and Ogrodniczuk and Woliński, M. (eds): *Wzbogacony korpus Słownika frekwencyjnego polszczyzny współczesnej*. Katedra Lingwistyki Formalnej, Wydział Neofilologii Uniwersytetu Warszawskiego, http://clip.ipipan.waw.pl/PL196x?action=AttachFile&do=view&target=operacje.pdf.

Ogrodniczuk, M. (2012): The Polish Sejm Corpus. In: *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pp. 2219-2223.

Okazaki, N. (2007): CRFsuite: *A fast implementation of conditional random fields (CRFs)*. URL: http://www.chokkan.org/software/crfsuite/.

Oravecz, Cs. and Dienes, P. (2002): Efficient Stochastic Part-of-Speech Tagging for Hungarian. In: *Proceedings of the 3rd LREC Conference*, Las Palmas, pp. 710-717.

Padró, L, Collado, M., Reese, S., Lloberes, M. and Castellón, I. (2010): FreeLing 2.1: Five Years of Open-Source Language Processing Tools. In: *Proceedings of 7th Language Resources and Evaluation Conference (LREC 2010), ELRA* La Valletta, Malta. May, 2010.

Petrov, S., Das, D. and McDonald, R. (2012): A universal part-of-speech tagset. In: *Proceedings of the 8th Language Resources and Evaluation Conference (LREC 2012)*.

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

Radziszewski, A (2013): A tieredCRF tagger for Polish. In: In: Bembenik, R. et al. (eds.): *Intelligent Tools for Building a Scientific Information Platform*. Berlin, Heidelberg: Springer, pp. 215-230.

Radziszewski, A. and Acedański, Sz. (2012): Taggers gonna tag: an argument against evaluating disambiguation capacities of morphosyntactic taggers. In: Sojka, P., Horák, A., Kopeček, I. and Pala, K. (eds.): *Proceedings of the 15th International Conference TSD, Brno, September 2012*. Heidelberg: Springer, pp. 81-87.

Radziszewski, A. and Śniatowski, T. (2011): A memory-based tagger for Polish. In: *Proceedings of the 5th Language & Technology Conference, Poznań*.

Ratnaparkhi, A. (1996): A maximum entropy model for part-of-speech tagging. In: *Proceedings of the conference on empirical methods in natural language processing* vol. 1, pp. 133-142.

Silfverberg, M., Ruokolainen, T., Lindén, K. and Kurimo, M. (2014): Part-of-Speech Tagging using Conditional Random Fields: ExploitingSub-Label Dependencies for Improved Accuracy. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pp. 259-264.

Spoustová, D., Hajič, J., Raab, J., Spousta, M. (2009): Semi-supervised training for the averaged perceptron POS tagger. In: *Proceedings of the 12th Conference of the European Chapter of the ACL*, pp. 763-771.

Straka, M., Hajič J. and Straková, J. (2016): UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. In: *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slovenia*, pp. 4290-4297.

Sutton, C. and McCallum, A. (2011): An introduction to conditional random fields. In: *Foundations and Trends in Machine Learning* vol. 4, no. 4, pp. 267-373.

Toutanova, K., Klein, D., Manning, C.D. and Singer, Y. (2003): Feature-rich part-of-speechtagging with a cyclic dependency network. In: *Proceedings of the 2003 Conference of theNorth American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pp 173-180.

Tufiş, D. (1999): Tiered Tagging and Combined Classifiers. In: *Text, Speech and Dialogue, Lecture Notes in Artificial Intelligence* vol, 1692, pp. 28-33.

Tufiş, D. and & Dragomirescu, L. (2004): Tiered Tagging Revisited. In: *Proceedings of the 4th LREC Conference*, pp. 39-42.

Turian, J., Ratinov, L.A. and Bengio, Y. (2010): Word representations: A Simple and General Method for Semi-Supervised Learning. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden*, pp. 384-394.

Woliński, M., Głowińska, K. and Świdziński, M. (2011): A preliminary version of Składnica – a treebank of Polish. In: Vetulani, Z. (ed.): *Proceedings of the 5th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pp. 299-303.

Woliński, M., Miłkowski, M., Ogrodniczuk, M, Przepiórkowski, A. and Szałkiewicz, Ł. (2012): PoliMorf: a (not so) new open morphological dictionary for Polish. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012*, pp. 860-864.

Wróblewska, A. and Przepiórkowski, A. (2014): Projection-based Annotation of a Polish Dependency Treebank. In: *Proceedings of the 9th International Conference on Language Resources and Evaluation, Reykjavík*, pp. 2306-2312.

Yang, Zh., Salakhutdinov, R. and Cohen, W. (2016); Multi-Task Cross-Lingual Sequence Tagging from Scratch. In: *Computing Research Repository*. URL: http://arxiv.org/abs/1603.06270.

Zipf G.K. (1935): *Psycho-Biology of Languages: An Introduction to Dynamic Philology*. Boston: Houghton-Mifflin.