Fine-tuning Chinese Whispers algorithm for a Slavonic language POS tagging task and its evaluation

Łukasz Degórski

Institute of Computer Science, Polish Academy of Sciences
ldegorski@ipipan.waw.pl

Abstract

Chris Biemann's robust Chinese Whispers graph clustering algorithm working in the Structure Discovery paradigm has been proven to perform good enough to be used in many applications for Germanic languages. The article presents its application to a Slavonic language (Polish), focusing on fine-tuning the parameters and finding an evaluation method for POS tagging application aiming at getting a very small (coarse-grained) tagset.

Keywords: unsupervised POS tagging, Chinese Whispers algorithm, slavonic languages, Polish

1. Motivation

Various language tools are based on supervised machine learning algorithms. They need training sets of data annotated manually. This is not a problem for popular languages and in general texts, as we can rely on the existence of relevant annotated corpora. However, for less popular (or funded) languages and dialects, and for specific narrow domains, we may need alternative solutions. Unsupervised machine learning comes to aid here, as all it requires is a reasonably sized collection of texts.

POS tagging is the necessary first step to creation of a knowledge-free and language-indepenent linguistic annotation chain (see Structure Discovery Paradigm, Biemann, 2012). For that reason we decided to begin with adapting the specific unsupervised methods to a morphologically rich, free word order language – Polish, to get an idea of the prospects of creating more advanced sets of tools for this and similar languages.

2. Goal

The goal was to identify the parameters that would allow us to generate automatically a clustering that would relate best to the traditional division into parts of speech. By that we mean

- not more than 7-10 reasonably-sized classes
- at least one (and preferably exactly one) class for nouns, verbs, adjectives, adverbs and other well defined parts of speech
- most words classified as members of one of these classes (short tail)

Of course this is just the target and we were sure we were going to miss it by a few inches. All we do have is words, co-occurrence relations and the Distributional Hypothesis, stating that words occurring in the same contexts tend to have similar meanings (Harris, 1954).

To illustrate the vagueness: it might be quite hard to tell an adjective from a noun, as adjectives and nouns in the same case have very similar distributional patterns. The distinction between adjectives and adjectival participles is more of a genetic than distributional nature, and as such it is even harder to grasp.

We deliberately did not use features other than co-occurrence, such as – for instance – word endings. The purpose was to stay as close as possible to the pure knowledge-free approach, allowing to generalise the conclusions for other similar languages without a need to fine-tune.

3. Related work

As Biemann says, "As the virtue of unsupervised POS tagging lies in its possible application to all natural languages or domain-specific subsets, it is surprising that in most previous works, only experiments with English are reported" (Biemann, 2012). He mentions just one work using languages from Slavonic, Finno-Ugric and Romance families – (Clark, 2003). Clark however used just one relatively short text (Orwell's *1984* translations from MULTEXT-East), combined both distributional and morphological information, and used an "extremely fine-grained" tagset, while we aim at getting an extremely simple tagset and want to use as little language information as possible.

4. Implementation

A set of input texts was concatenated to form one big file. This file was normalised, i.e. converted to lowercase, with special characters and interpunction replaced by symbols like `[crlf]` or `[comma]`. Frequencies were then counted for each character/symbol.

Then, the method described in (Biemann, 2006b) could be applied. We just focused here on what he calls "partitioning 1" (just for high frequency words):

- determine f feature and t target words from the frequency counts mentioned above (i.e. take top f words

Study was financed by research fellowship within Project „Information technologies: Research and their interdisciplinary applications", agreement number UDA-POKL.04.01.01-00-051/10-00.

as the feature words list, and take top t words as the target words list),

- construct the graph from context statistics, i.e. add an edge of weight $\frac{100}{1-cos(a,b)}$ between a and b when this weight exceeds the similarity threshold s; $cos(a,b)$ is a cosine of the co-occurrence vectors representing a and b, generated in regard to the feature words list,
- apply the Chinese Whispers clustering algorithm (Biemann, 2006a) to the created graph.

Chinese Whispers is a "very basic – yet effective – algorithm to partition the nodes of weighted, undirected graphs". Its input is a graph, and its output – a clustering of this graph. At the beginning each node gets a different class. Then in a small number of iterations (we used 20) the nodes are processed in a randomised order so that each node inherits the strongest class in the local neighbourhood. Formally the algorithm does not converge, so it is heuristic in a way, but nonetheless it serves its purpose – clusters the graph quickly and without preassuming the target number of clusters.

What was the most interesting for us was the optimal configuration of parameters that would fit best the Polish language and the defined task. In addition to f and t the investigated parameters included the radius of the context window w (1, 2 or 3; the bigger the window, the longer the co-occurrence vector – its length is $2 \cdot w \cdot f$) and similarity threshold s.

A Java application was written to generate a graph (in the form of two text files, listing nodes and edges with weights) for each analysed configuration. The Chinese Whispers algorithm in the original Java implementation was then applied to the graph three times, as the clustering algorithm itself also has a parameter, specifying how labels propagate in the graph (*top*, *dist_log* and *dist_nolog*).

5. Evaluation

The configurations have been evaluated in a supervised way. This should not be confused with the way the target algorithm works. The evaluation aimed at determining the optimal configuration of the unsupervised Chinese Whispers machinery.

5.1. Corpus

The evaluation corpus was constructed by taking some texts (representing fiction, non-fiction and newspapers) from the National Corpus of Polish (Przepiórkowski et al., 2012). Its size was 37 million words.

At the beginning a 5-million-word fiction corpus was used, but the experiments with the bigger one have shown that 5 million might be not enough to get a grasp of enough distributional dependencies – the results seemed much more random.

Planned experiments with 200 million words failed for technical reasons.

5.2. Procedure

For evaluation we used the Morfeusz morphosyntactic analyser (Woliński, 2006). It gives a POS tag or a few tags for each word. There is no disambiguation, as it is not a tagger, and context-aware tagging would not make sense when what we deal with is a list of separate words. Thus we accept all Morfeusz tags as correct, so the same word might score in the same cluster as a noun and as a verb. To illustrate the scale of this phenomenon: we got up to 4% more tags than words.

For each cluster and POS tag (one from the arbitrarily selected list of basic traditional parts of speech: verb, noun, adjective, adverb, personal pronoun, numeral) the "cluster-as-POS" score was determined. For instance, to determine the cluster-as-verb score we looked at precision

$$\frac{\texttt{number of verbs in the cluster}}{\texttt{cluster size}}$$

recall

$$\frac{\texttt{number of verbs in the cluster}}{\texttt{number of verbs in all clusters}}$$

and used $F_{\beta=2}$-measure to weigh precision two times higher than recall. Then the best among ratings was chosen and if, for instance, the verb score was the highest, the cluster was marked as a class of verbs.

Afterwards, average scores were counted separately for all six types of clusters (average verb score, average noun score, ...) and added up – with weights, so that the noun score is much more important for the final result than the numeral score. The weights are normalised and have been chosen according to the proportions of POS tags in the manually annotated million-word subcorpus of the National Corpus of Polish:

Part of speech	Weight
Verb	0.248
Noun	0.459
Adjective	0.179
Adverb	0.059
Personal pronoun	0.032
Numeral	0.022

Table 1: Normalised POS weights used in evaluation

Consequently, the weighted scores may still be interpreted as F-measure values. In a perfect clustering we would get six clusters – one of each type. If the verbs (for example) get dispersed among two clusters, each of them will have lower recall, so the average verb score will fall. Non-verbs in a verb cluster will obviously lower the verb precision. Failing to identify a less important class at all (e.g. when all personal pronouns get classified as nouns) will not affect the outcome a lot, although it will intervene twice: as zero in pronoun score and in lower noun score (because of falling noun precision).

6. Results

6.1. Parameters

Each experiment was defined by four context matrix construction parameters (1-4) and one Chinese Whispers application parameter (5):

1. **Number of feature words**. Feature words are the most frequent words in the corpus. Similarity of target words (i.e. those classified) is measured by looking at their patterns of co-ocurrence with the feature words. We would expect the feature words to be function words.

2. **Number of target words**. The classification (clustering) of target words is the output of the algorithm. Not all target words will be classified, especially in configurations with higher edge weight threshold, as vertices are ignored when they are not connected with at least one edge with weight above the threshold.

3. **Window radius**. Window radius of 2 means that the pattern of co-ocurrence of target word T with feature word F is a 4-element vector (number of occurrences of T two words before F, one word before F, one word after and two words after F). We tested window radii 1, 2 and 3. The bigger the radius, the longer it takes to compute cosine similarity and find out the weight of the edge between two target words. The positive influence of increasing the window radius is not as obvious as it might seem. Indeed, setting it to 1 unduly promotes clusters like "words that often follow the preposition *po*", but the difference between 2 and 3 is not that clear any more.

4. **Weight threshold**. Edges with low weights tend to introduce noise, as their distributional similarities may be purely random. On the other hand, setting the threshold too high reduces the number of classified target words, because vertices with no edges are not taken into account.

5. Chinese Whispers **label propagation algorithm**. One of *top*, *dist_log* and *dist_nolog*. The *top* tends to render fewer bigger clusters than the other two.

6.2. Configurations

To make experimenting with so many parameters feasible, we created a few basic configurations to focus on:

Conf.	FWs	TWs	Radius	Threshold
1	110	4200	3	150
2	200	4200	2	150
3	110	10000	2	150
4	200	10000	3	500

Table 2: Initially tested configurations

After initial experiments it turned out that 4200 target words is not enough, and that the noise makes threshold of 150 too small. In the following experiments, only configuration #4 and its modifications #5, #6 and #7 were used. At the very end one more was added (#8).

Conf.	FWs	TWs	Radius	Threshold
5	200	10000	2	500
6	200	10000	3	300
7	200	10000	1	500
8	200	25000	3	500

Table 3: Configuration modifications

With that many target words and a big radius #8 took quite a long time to compute, but was interesting as it classified much more words than the other configurations.

Each configuration was tested with all three label propagation algorithms, so they are not treated as a part of configuration definition.

6.3. Derivatives and modifications

The experiments were conducted in a tree-like fashion, interesting configurations were modified a little to create derivatives, while those that seemed uninteresting were discarded. Only #4, #5, #7 and #8 and their derivatives got to the final stages. All of them had the threshold of 500; the derivatives with thresholds 150, 1000 and 2000 were also investigated.

A certain structural modification was also tested. As the effectiveness for Polish was more important than perfect purity of the method, we decided to check how editing the feature word list might affect the results. In a pure knowledge-free approach we took the most frequent n words ($n = 200$ in all final basic configurations) without analysing the list in any way. The practical approach should be language annotation-free, but not necessarily language information-free (as the basic goal of the whole research is to find the best configuration for the Polish language). Thus we spared ten minutes to remove the content words (those with a statable lexical meaning) from the frequency list, trying to leave only function words, and then took the top 200. About 15% of words from top 200 and about 30% of words from top 370 have been removed in this operation. Modified configurations have been marked with an apostrophe, e.g. #5'. Modified configurations have the same number of target words, just the list is a little bit different.

The list has been cleaned up to position 370 (further down the list the function words seemed to be much sparser), so it seemed natural to check the chosen configurations also with a longer feature word list of 370 words. Such configurations were marked by adding 10 to the number.

6.4. Hierarchical agglomeration

Practically all clusterings turned out to have very long tails – for instance, half of the noun-like words were grouped in one cluster, and the rest was dispersed among dozens of smaller clusters. While those specific clusters might be also interesting for other applications (a lot of them had sensible semantic interpretations), it was disastrous for coarse-grained POS-tagging. For this reason we wanted to find a way to agglomerate similar clusters in the output of Chinese Whispers algorithm.

id	Classified words	Output clusters	$F_{\beta=2}$	# FWs	# TWs	Radius	Threshold	Agglomeration alg. sequence
w1	3086	7	57.4%	370’	10000	3	1000	t, t, l
w2	2521	2	61.7%	370’	25000	3	2000	t, t, t or n
w3	8992	2	51.2%	200’	10000	1	500	t, l, t
w4	19412	2	49.9%	200’	25000	3	500	t, t, n
w5	8597	8	48.8%	370’	10000	1	500	t, l, l

Table 4: Summary of the best configurations. Note that all of them use modified feature word list, hence the ’ mark.

Two methods have been tried. The first was using standard hierarchical grouping algorithms implemented in the R statistical package. The results were very discouraging. The more natural way seemed to be running Chinese Whispers again, treating the clusters we got in the first stage as input, i.e. as next level “hypernodes”, connected with “hyperedges”. The weights of the newly created hyperedges were computed by summing the weights of all edges connecting the first stage nodes forming the second stage hypernode (Biemann, 2012).

In this way we got “hyperclustering” into much less classes. The results in terms of our evaluation were much better, but still not good enough. It seemed natural to repeat the operation, creating third stage “hyperhypernodes” with appropriate edges connecting them, and running Chinese Whispers one more time. This step reduced the number of classes furthermore, in case of some configurations to a value close to the number of parts of speech we intended to extract.

These steps rendered dozens of results, because in each agglomeration level we get three times more for each configuration. Before agglomeration there were 3 results for each, for Chinese Whispers propagation parameter set to top (*t*), dist_log (*l*) or dist_nolog (*n*). At the last stage we got to 9 possible combinations, so an evaluated configuration could be described for instance as:

```
#17’-1K-(t,n,n)
```

meaning: configuration 7 with a modified feature word list of length 370 and threshold 1000 (370/10000/1/1000), agglomerated twice (initial run with top, then agglomeration with dist_nolog and second agglomeration with dist_nolog).

6.5. Winners

The choice of an optimal configuration depends on the intended application, the best ones are presented in Table 4. If we are happy with a comparably low number of classified words, the winner would be one of the high-threshold configurations: *w1* or *w2*. To get more classified words, we have to lower the threshold and might go for *w3* or *w4*. However the last two classifications (and the top-scoring *w2* as well) only recognise two clusters: nouns and verbs. If we want more classes, *w5* seems to be the best, as its output is much more balanced: eight clusters, identifying five parts of speech (no personal pronouns).

7. Conclusions and future work

The chosen evaluation method turned out to be properly balanced with regard to the levels of tagging finegrainedness it promotes: among top-scoring configurations there were both the one recognising 5 categories and the one that classifies all words as nouns or verbs. The choice depends on the application: if we just want to identify nouns, a setting rendering just two classes, one of them recognising this part of speech with precision and recall around 75%, would be much better than identyfying all 6 classes with much lower results for nouns in particular.

While the F-score of about 50-60% might seem low in general, we consider it quite good for a knowledge-free approach not needing any information other than a corpus of texts in the language and a hint what is and what is not a function word in this language, especially when dealing with a language with a complicated morphology and lower token/type ratio. Biemann and Clark do not use F-measure, but (Biemann, 2012) compares his and Clark’s results with regard to V-measure (Rosenberg and Hirschberg, 2007) that might be considered analogous as it is a harmonic mean of homogeneity and completeness, values comparable to precision and recall, and takes values between 0 and 1. Those results reach 0.708 for English and 0.684 for German.

The manually updated information on what is a function word turned out to be quite important – almost all top-scoring configurations were those with the modified list of feature words.

For the defined task two-level hierarchical agglomeration is a must. The best score after the first level of agglomeration was 37.7% (and only for a rather useless configuration with a very high threshold), and without any agglomeration it rarely exceeded 10%.

The choice of the label propagation algorithms in Chinese Whispers is important. For instance, for the same #17’ configuration with two levels of agglomeration the score varied from 28.6% to 48.8% depending solely on this choice in each of the three runs.

Future work should begin with collecting different texts and determining the stability of the way particular configurations behave – whether the 37-million word corpus was enough to represent Polish language in general or not. There is also a lot of space to play more with the parameters, for instance to try to discover a configuration that could classify about 20000 words and recognise more than verbs and nouns. The other solution would be to implement the Biemann’s algorithm for medium and low-frequency words, as here we just focused on the high-frequency words.

It might be also interesting to try working with a much bigger corpus, for instance 200 million words. That would require rewriting the Java program to make it need less resources, or using a machine with huge amounts of RAM.

8. References

Biemann, Chris, 2006a. Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1. Stroudsburg, PA, USA: Association for Computational Linguistics.

Biemann, Chris, 2006b. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the COLING/ACL-06 Student Research Workshop*. Sydney, Australia.

Biemann, Chris, 2012. *Structure Discovery in Natural Language*. Berlin: Springer.

Clark, Alexander, 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1*, EACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics.

Harris, Zellig, 1954. Distributional structure. *Word*, 10(23):146–162.

Przepiórkowski, Adam, Mirosław Bańko, Rafał L. Górski, and Barbara Lewandowska-Tomaszczyk (eds.), 2012. *Narodowy Korpus Języka Polskiego*. Warsaw: Wydawnictwo Naukowe PWN.

Rosenberg, Andrew and Julia Hirschberg, 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning(EMNLP-CoNLL)*.

Woliński, Marcin, 2006. Morfeusz — a practical tool for the morphological analysis of Polish. In Mieczysław A. Kłopotek, Sławomir T. Wierzchoń, and Krzysztof Trojanowski (eds.), *Intelligent Information Processing and Web Mining*, Advances in Soft Computing. Berlin: Springer-Verlag, pages 503–512.