Named-Entity Chunking for Norwegian Text using Support Vector Machines

Bjarte Johansen
Department of Information Science and Media Studies
University of Bergen
Postbox 7802, N-5007 Bergen, Norway
bjarte.johansen@uib.no

October 16, 2015

Abstract

Named-Entity Chunking is part of the Named-Entity Recognition (NER) process and is the task of identifying which parts of a text are names. This task is usually done as an implicit part of the recognizer, but because previous attempts at NER for Norwegian text focus only on the recognition, this research represents an attempt to develop an explicit chunker. The research shows that if we only focus on demarcating names and not on discovering their type as well, we are able to accurately (>95% F_1-score) find the names in Norwegian text using Support Vector Machines.

1 Introduction

Named-Entity Chunking (NEC) is the task of demarcating which segments of a text are parts of a named entity and which are not. A named entity is a specific person, place, event, etc. Chunking is different from the process of Named-Entity Recognition (NER) in that we are not interested in the type of the entity, but only finding the entities themselves.

Maurits	Escher	was	a	Dutch	artist	.

Table 1: An example sentence.

To understand better what the problem is we can look at the example in table 1. In the example, we see *Maurits Escher*. When we are chunking the sentence we are trying to discover that the terms *Maurits* and *Escher* are part of the same named entity.

We also see the term *Dutch*, and even though it refers to a nation, we do not consider it by itself an entity. *Artist* also appears, which on the other hand is an entity, but it is a general category and does not refer to a single individual (or thing) and is therefore not a *named* entity.

This paper was presented at the NIK-2015 conference; see http://www.nik.no/.

The reason for developing an explicit chunker instead of as part of a NER tool is that previous research on recognition for Norwegian focused on only categorizing the names in pre-chunked data (Haaland, 2008; Nøklestad, 2009). The exception is Jónsdóttir (2003), who do chunking as a part of the recognizer. In future research, we are also interested in collecting entities from newspaper articles, but are not necessarily interested in the *type* of the entities.

It is important to do this type of research for languages like Norwegian as it shows that these types of methods used in this research generalizes over different languages given the right feature selection and training material.

This paper proposes a solution to the problem of named entity chunking in the context of Norwegian text. It does that by:

1. Discussing and identifying some characteristics of Norwegian that are important for named-entities in section 3.
2. Running experiments to show that Support Vector Machines (SVM) are able to, given the features that we have defined, provide state of the art performance on this task. See section 4.
3. Comparing our solution to what others have done before in section 5.
4. Concluding on the results and discussing any future work in section 6.

2 Related work

Jónsdóttir (2003) did some early work on chunking and recognition for Norwegian. They used a ruled-based approach through the use of constraint grammar. The approach did provide good recall scores (>90%) for NER, but the precision did not reach satisfactory results (<50%). Jónsdóttir does not provide the corresponding numbers for their NEC.

Nøklestad (2009) and Haaland (2008) also worked on NER for Norwegian texts. Nøklestad uses a Memory-Based Learning approach while Haaland uses Maximum Entropy Models. The main problem the approach of both Nøklestad and Haaland is that they are dependent on previously chunked text to work correctly. The chunker they used is however no longer available.

Zhou and Su (2002) do NER through a Hidden Markov Model based chunker. This is what inspired us to develop an explicit chunker separate from the recognizer. Their paper is however only for English.

Kudo and Matsumoto (2001) does use SVMs to identify English base phrases, where named entities would be one such base phrase, and do very well. However, they do not try to extend their work to other languages, which is what we are doing in this paper by trying to use the same method for NEC of Norwegian text.

3 Important characteristics of Norwegian text

In this section we describe some characteristics that are important for named entities in Norwegian text.

Capitalization

In the book "Skriveregler" (translation: rules of writing) Vinje (1998) presents 19 rules for capitalization of words in Norwegian. The most notable are however the rules for titles and organizations. Titles are never capitalized in Norwegian; kong

Harald (King Harald). The names of organizations should, as a rule, only have the first term capitalized. E.g "Den norske stats oljeselskap" (Statoil). However, there are many exceptions to this rule. This means capitalization is not always a good indicator for Norwegian names. Haaland (2008) provides a summary of the rules in English.

Compound words

In Norwegian there is usually a semantic difference between two sentences if you use a compound word or two separate words, e.g. "røykfritt" vs. "røyk fritt" Språkrådet (2009). The first translates to "no smoking" while the second to "smoke freely". This affects grammar, but also named entities. Organizations can sometimes consist of compound words like "Luftforsvaret" (The Air Force). If we had instead written "Luft forsvaret" it would instead translate to "Air out the Armed Forces". (The capitalization of "forsvaret" is correct in this instance if we are referring the Armed Forces in general and not the institution.)

Polysemy

Norwegian also has many polysemes; words that mean different things in different contexts. An example would be a a word like "historie" which could both be translated to "story" or "history" depending on the context Jónsdóttir (2003). There are also names in Norwegian that are polysemic and can be quite difficult to understand without a wider context. The sentence "Bjørn er farlig" can be translated to both "Bears are dangerous" or "Bjørn is dangerous" as Bjørn can be the given name of a person. It could therefore be important to capture some of the context to disambiguate between the terms which are part of a name and those that are not.

Two written forms

Norwegian has two written forms; nynorsk and bokmål. They vary slightly in grammar and spelling, but are otherwise quite similar. The variance between the written forms does, however, mean that we cannot just run a chunker that has been trained on one of the forms and expect it to work well on the other. In this paper we only focus on bokmål as it is the most popular written form.

Regional variances

Norway has many dialects and the dialects can affect the spelling, grammar and choice of words within the same written form. F.ex., normally, Bokmål and nynorsk both have three grammatical genders; female, male and, neutral. An exception is the Bergen dialect which only uses two grammatical genders; common and neutral (Bordal, 2015). A name like "Tariffnemnda" (The Tariff Committee) could therefore also be spelled "Tarriffnemnden" depending if the writer speaks the Bergen dialect or not. This does not have an immediate effect on chunking, but increases the number of different values in the data set and therefore makes it more difficult to learn from if it is not handled properly.

4 Experiment

During the research for this paper we did multiple experiments to test whether a difference in how the terms are categorized affects the accuracy of the classifier. We

chose to use a SVM classifier as they have shown themselves to work well within text classification (Joachims, 1998).

Support Vector Machines

A SVM is a supervised learning algorithm "where the goal is to find a decision boundary between two classes that is maximally far from any point in the training data" (Manning et al., 2008, p. 293).

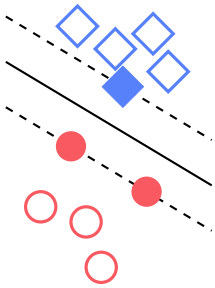

Figure 1: Example result of training a SVM.

In figure 1 we can see an example model after training a SVM with samples from two different classes. The hyperplane is the solid line in the middle, while the stippled lines is the margin to the hyperplane. The solid-coloured samples on the margins are the support vectors of the model. The goal of training a SVM is to find the hyperplane with the largest margin between data points in two different classes.

For the research in this paper we were interested in distinguishing between multiple classes, but traditionally a SVM is only able to differentiate between two. To get around this constraint we use an extension to SVMs which supports multi-class data; the "one-versus-one" approach. The classifier builds a SVM for each pair of classes and chooses the class that is selected by the most classifiers.

In the case of labeling errors there might not be possible to find a hyperplane that cleanly separates the different classes. To get around this constraint we use the soft-margin method which allows some classification errors. C is the soft-margin parameter to the error function and controls how much a classification error is penalized (Vert et al., 2004). The size of C can therefore result in over- or under-fitting by making the SVM choose a small or large margin hyper-plane.

The kernel type that we used was the radial basis function (RBF) which allows the SVM to also classify nonlinear data by lifting the data into higher dimensions were they might be linearly separable after all. γ is the free variable of the kernel and decides how the points in the problem space are lifted into higher dimensions to make it easier to separate the different classes from each other. The RDF should be able to find any linear separation that both a linear and polynomial kernel is able to find, though it is more expensive to compute. The implementation we used was the libsvm library (Chang and Lin, 2011).

Tagging

(Kudo and Matsumoto, 2001) presents 5 main ways of tagging chunks in text. For the experiment in this paper, we chose the IOB2 method of tagging to demarcate the named entity chunks. It uses 3 tags to identify tokens

I A token inside a chunk.
O A token outside any chunk.
B A token at the beginning of a chunk.

The reason for choosing the IOB2 tagging scheme is because it provides more of each tag in the data set than the other systems, which makes it easier for the SVM training algorithm to learn from the data set. We could also have used the IOE2 scheme; which is equivalent to IOB2, but tags the end of chunks instead of the beginning.

The other tagging methods either only tags the ends/beginning of tags between the boundary of two chunks (IOB1/IOE1) or introduces extra tag(s) (Start/End) and therefore leaves fewer instances of some tags and gives the SVM little material to learn from.

Feature vector

In the example sentence in table 2 we can see that each word is accompanied by its part-of-speech (PoS), lemma (the canonical form of a word) and, direct translation. These data are the result of running the sentence through the OBT.

PoS	noun	verb	det	noun	clb
Sentence	Industrien	har	ingen	problemer	.
Lemma	industri	ha	ingen	problem	.
Translation	The_industry	has	no	problems	.

Table 2: Example PoS, sentence, lemma and, direct translation.

The data are used to transform each word in the sentence into a feature vector that can be used to learn a model for the data or predict new instances. In table 3 we can see the features that has been selected together with an example from one of the words in the sentence in table 2.

Features	Vector
lemma - 2	industri
lemma - 1	ha
lemma	ingen
lemma + 1	problem
lemma + 2	.
PoS - 2	noun
PoS - 1	verb
PoS	det
PoS + 1	noun
PoS + 2	clb
Capitalized?	0
Last 4	ngen

Table 3: The defined features and example vector.

The features that we chose were the lemma of the current word and the 2 words on each side, the PoS of each word, if the word is capitalized and, the 4 last characters of the word.

We constrained the use of the surrounding words to two words on each side. This was because we wanted to capture some of the context for each word, but at the same time we wanted to keep the cost of training the model at a level that was possible within the available resources. We also chose to use the PoS for each word as the PoS contains hints to whether a word is part of a name or not. F.ex would a name (often) be classified as a noun by the OBT.

We use capitalization as a feature because we want to downcase the lemmas to keep the number of possible values low. The reason we keep the last 4 characters of the word is because of a feature of Norwegian last names: Many Norwegian last names has the same ending, like Johansen and Evensen or Fjellheim and Norheim.

Setup

We use the same data set as Nøklestad (2009) and Haaland (2008) where the terms are tagged with their type (person, organization, etc.) and their grammatical class. Since the grammar in the data set was tagged with an older version of the Oslo-Bergen Tagger (OBT), we cleaned the data and align the tags with the new version of the OBT. The OBT "is a robust morphological and syntactic tagger developed at the University of Oslo and at Uni Computing in Bergen (Tekstlaboratoriet and Uni Computing, 2014)."

An overview of the data set for this research is available in table 4. It consists of 210 newspaper articles, 46 magazine articles and 9 works of fictions with a total of 230453 tokens (words, punctuation, symbols, etc.). There are a total of 7505 entities in the data set.

Resource	Sources	Tokens	Entities
Newspaper articles	210	107814	4474
Magazine articles	46	63763	1916
Works of fiction	9	58876	1115
Sum	265	230453	7505

Table 4: Description of data set.

We tagged each token in the data set with the IOB2 tagging scheme. The categories with the number of tags are specified in table 5.

Category	Count	Percent
(B)eginning	7505	3.26%
(I)nside	2583	1.12%
(O)utside	220365	95.62%
Total	230453	100.00%

Table 5: Number of terms in each category.

Each token, together with the surrounding context, was then transformed into a feature vector.

The SVM library we used for this research does not support a string vector as input, only a sparse numerical matrix, so we built a tool to convert between our text vectors and this format.

To test different parameters to the SVM learning algorithm we did a grid search over a set of variables. For the C value we used the range $2^{-5...15}$ where the power increments by 2 at each step.

For γ we used the range $2^{3...-15}$ where the power decrements by 2 at each step.

For every parameter option we did a 5-fold cross validation to check the result of the learned model. To be able to accurately test the classifier after it had been learned, we also removed 20% of the training data to use for testing by randomly selecting 20% of the instances from each class. The reason for selecting randomly from each class instead of the total data set was to avoid randomly selecting only the dominating class, Outside, and ensuring that we had enough test instances for each class.

To distribute the calculations over many machines and improve the time it takes to test all combinations of C and γ we used GNU Parallel (Tange, 2011); a shell tool for executing jobs in parallel.

Results

The results from the experiment are shown in table 6. There we can see that the chunker has a higher precision than recall. This means that the chunker is usually correct when it reports that it has found a chunk, but that it is not able to find all chunks in a text. However, the F_1-score tells us, as the harmonic mean between the precision and recall, that the accuracy of the NEC is quite good.

Precision	Recall	$F_{\beta=1}$
97.95	95.34	96.63

Table 6: Results of experiment.

To calculate the precision and recall of the system we only looked at chunks that are an exact match to the corresponding chunk in the data set. A partial match is therefore not only a false negative, but also a false positive, as the exact chunk found by the system is not found in the original data set.

We also discovered that if we removed capitalization as a feature from the training data, the recall of the chunker dropped significantly (<50%), but the precision stayed high (>95%). Taking into consideration the many capitalization rules of Norwegian, we believe this result shows that while capitalization is important for finding the start of entities, it is not as important for the following parts of the entities. We can say that because the precision is still high, meaning that the entities that the chunker does find when capitalization is removed from the features are mostly correct.

If we also look at the entities the chunker that produced the results in table 6 classified incorrectly it seems like it has problems with names that are at the beginning of sentences. This might be because of the polysemy of certain names in Norwegian and that capitalization in words at the beginning of sentences is not a good indicator for a name since (almost) all sentences begin with a capitalized word.

5 Discussion and Conclusion

If we compare our system to the CONLL shared task in 2000, we can see that the score for the system is significantly better than the baseline precision, recall and F_1-score (Tjong Kim Sang and Buchholz, 2000). The system even performs

better than the best performing chunker from that competition (95.8%) (Kudo and Matsumoto, 2001). However, this is not a completely fair comparison as they are trying to solve *any* text chunking problem for English: Noun phrases, verb phrases, adjective phrases, etc. It is also focusing on English and not Norwegian. Despite this, we can use the number as a baseline for how a chunker should perform and can therefore conclude that our chunker is doing quite well.

Zhou and Su (2002) does equally well on chunking names in English text as we do on chunking Norwegian text with an F_1-score of 96.6%. However, their data set contains only 1330 instances, and it is therefore difficult to judge the generality of their chunker. Though the data set used for training the chunker in this paper is only moderately sized it is still over 5 times as big at 7465 entities.

Previously reported research on named entities in Norwegian text focuses only on NER and not on NEC, but to compare our research to theirs we reduce their system to a binary "if they find a name or not" and look only at the total F_1-score of the whole system. We do this so we can tell if our chunker is better at finding names than a combined chunker and recognition approach.

Comparing this research to the research from Nøklestad (2009) and Haaland (2008) is not completely appropriate as their research only works on pre-chunked text. They need the names in the text to be already picked out and then uses the surrounding context to discover the type of entity.

The only candidate that we know of that goes from untagged Norwegian text to NER is the work by Jónsdóttir (2003) and they report a precision of 45%, recall of 92% and a final F_1-score of 60%. If we compare this score with the F_1-score of our chunker we can see that our chunker is more precise (98%), offers better recall (95%) and, is therefore also more accurate (97%). However, this comparison is also problematic since theoretically their chunker could perform perfectly and the loss in precision is from the recognizer. We still provide the comparison as it is the only work on Norwegian text that is close to our research.

From the results we see that by using SVMs we are able to accurately (>95% F_1-score) find names in Norwegian text and that if we are interested in finding *just* the names in a text and not their type, it is better to implement an explicit chunker.

6 Future work

As mentioned in section 2, there are several research papers describing approaches to NER, but they need the named entities in the text to be pre-tagged and will therefore not work with untagged data. A future path for this research would be to test the NEC developed in this project with these approaches and see if we are able to do streaming NER on live data.

Though we were able to identify some characteristics of Norwegian text that applies to the tasks of Named-Entity Chunking and Recognition, we have not been able to find a good way to include what we learned in the feature vector used to train the chunker we developed in this paper. We have seen that polysemy and the many capitalization rules of Norwegian can affect the accuracy of our research. In the future we should try to find ways to use these characteristics to improve the accuracy of the chunker.

References

Guri Bordal. Substantiv. `https://snl.no/substantiv`, July 2015.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Åsne Haaland. *A Maximum Entropy Approach to Proper Name Classification for Norwegian*. PhD thesis, University of Oslo, March 2008.

Thorsten Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.

Andra Björk Jónsdóttir. *ARNER, what kind of name is that? - An automatic Rule-based Named Entity Recognizer for Norwegian*. PhD thesis, University of Oslo, May 2003.

Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics, 2001.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

Anders Nøklestad. *A machine learning approach to anaphora resolution including named entity recognition, pp attachment disambiguation, and animacy detection*. PhD thesis, University of Oslo, June 2009.

Språkrådet. Orddeling og særskriving. `http://www.sprakradet.no/Vi-og-vart/hva-skjer/Aktuelt-ord/Orddeling-og-sarskriving/`, 2009.

Ole Tange. Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, 36(1):42–47, Feb 2011. URL `http://www.gnu.org/s/parallel`.

Tekstlaboratoriet and Uni Computing. The oslo-bergen tagger. `http://www.tekstlab.uio.no/obt-ny/english/index.html`, 2014.

Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, ConLL '00, pages 127–132, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. doi: 10.3115/1117601.1117631. URL `http://dx.doi.org/10.3115/1117601.1117631`.

Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. A primer on kernel methods. *Kernel Methods in Computational Biology*, pages 35–70, 2004.

Finn-Erik Vinje. *Skriveregler*. Aschehaug, 7 edition, 1998. Gjennomgått av Norsk språkråd og anbefalt for offentlig bruk av Kulturdepartementet.

GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics, 2002.