Improving chunker performance using a web-based semi-automatic training data analysis tool

István Endrédy

Pázmány Péter Catholic University, Faculty of Information Technology and Bionics
MTA-PPKE Hungarian Language Technology Research Group
50/a Práter Street, 1083 Budapest, Hungary
endredy.istvan@itk.ppke.hu

Abstract

Fine tuning features for NP chunking is a difficult task. The effects of a modification are sometimes unpredictable. Feature selection/tuning is usually made in trial-and-error style with long iterating times. Thus, an online toolkit was developed, which addresses three tasks: (1) it can investigate a training corpus made for NP chunking, (2) it makes POS feature suggestions for better NP chunking, and finally (3) the new dataset can be exported. The kit automatically counts an approximated F-score on the fly, as a quick feedback to the linguist. The kit was tested on English and Hungarian corpora. It proved to be able to accelerate preparing datasets for NP chunking effectively, and it gives useful POS feature suggestions from WordNet, resulting in better F-scores. The toolkit needs only a browser (no dependency, nothing to install), and it is easy to use even for non-technical users. The development of features can be controlled in a user friendly way. The tool combines the abstraction ability of a linguist and the power of a statistical engine.

1. Introduction

The task of noun phrase (NP) extraction from a sentence is called NP chunking. This can be considered a sequential tagging process which gives a label to each word describing its role in the phrase. Various label sets are used in this task, out of which the IOB set contains only three items: beginning of a NP ("B"), inside ("I") or outside the NP ("O"). One of these labels are assigned to each word in the text. The label set can also be more detailed. For instance, a chunk composed of one token (a single) may have its own label ("S").

The input data of chunking traditionally has the following format: each word is in a new line with a tab-separated feature list. Feature can be any property of the given word: part-of-speech category, length of the word, countable/uncountable, animate or not, abstract or not, etc. The linguist has to find adequate features, the use of which will result in better F-scores. In a typical scenario, when the selection or modification is done, the NP chunking process will be run and after a while its quality can be seen. If the improvement is only moderate or not satisfactory, the linguist can modify the parameters of the NP chunker and some input features, and has to execute the program again. This iteration, however, takes time due to the requirements of statistical learning methods.

The present article would like to help at that point where the linguist tunes the features. The toolkit described gives an overview about the structure of the training set, and can give a feedback about the modifications of features on the fly, without running real tests. Standard evaluations will be needed only at the end, since the toolkit is used only to prepare datasets and to speed up feature tuning.

Certainly, the method described (and its results) can be reached with command line tools as well. The added value is this user friendly online tool, which can be used for users without advanced computer skills as well.

2. Related works

There are two basic types of improving chunker performance: software or data improvements. Several researches create, combine or improve a tagger engine, which performs better on the same data (Koeling, 2000; Osborne, 2000; Sun et al., 2008). Other approach is to find, define or combine features and/or labels which produce better F-score for the given chunking task (Shen and Sarkar, 2005; Simon, 2013). F-score is typically counted against a gold standard: correctly recognized chunks are true positive (tp), incorrect ones are false positive (fp), and missing ones are false negative (fn). Precision is counted by tp / (tp + fp), recall by tp / (tp + fn), and F-score is the harmonic mean of precision and recall.

Specialization also works, for instance, when IOB labels are split into more detailed classes (Molina and Pla, 2002; Shen and Sarkar, 2005). In that approach labels (above a threshold) are completed with part-of-speech (POS) category. For example, the following (word, POS, IOB) tuple *(You, PRP, B-NP)* get detailed IOB label: *(You, PRP, PRP-B-NP)*. Our solution is similar, however, we try to fine tune the POS info instead of the labels.

For feature tuning the classical approach is the trial-and-error style: the linguist adds many features to the data (new columns to each word) and (s)he tries out them and their combinations. This means several train/test phases and more weeks iteration time. This paper is connected to the data improving, and it tries to help only to prepare the dataset before tagging.

3. Problem of feature tuning

A key aspect of NP chunking is the set of features which are used in labelling algorithms of any type. At this abstraction level, the feature set is independent from the chunking method. Thus, the task of feature selection can be separated from the actual algorithm, and its direct effect on F-score can be monitored.

Thus questions like how the features can be selected or which ones help labelling come up. Since in solving such problems, a linguist is usually involved, we might rely on their good intuitions. However, their work is also to be supported.

The available statistical NP chunker methods have a common point from this aspect (Koeling, 2000; Osborne, 2000). All of them are used as black boxes, no one can control their feature level processes. However, they can easily be applied to any language or data. On the contrary, rule based systems are fully controlled by linguists, but they are built manually in a slow process, and they are strongly connected to the given language and part-of-speech tagset (Déjean, 2000; Johansson, 2000).

4. Idea

The basic idea of our method is to combine the intuition and control of a linguist and the power of statistical engines. A set of texts contains useful information regarding the connection between features and IOB labels that can be extracted by statistical algorithms using these texts as training data. For instance, a human would hardly find a few features which behave differently with respect to IOB labelling. These features therefore need to be split into more types.

Our aim is to detect and show the best features (from the aspect of IOB labelling), and to find the ones which need manual fine tuning. At the end of the process, training and test sets can be exported (see Figure 1) and used in any type of NP chunkers. The features tuning in this article focuses only on the POS information. However, the tool can work on other features as well, but POS feature tuning is able to demonstrate the tool and its usage.

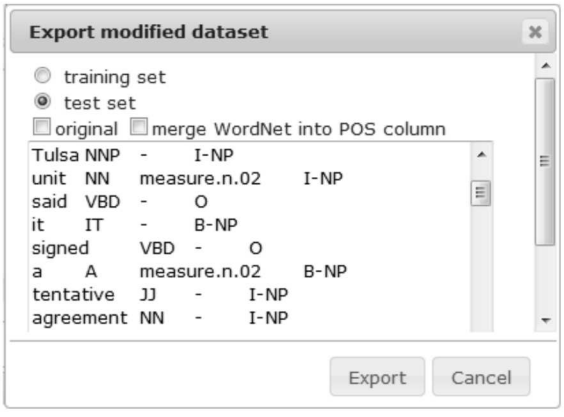

Figure 1: Dataset preview at exporting (word, POS, wordnet synsets, iob label)

5. Toolkit

An online tool[1] was developed, which contains a training and test set for NP chunking. At the moment CoNLL2000 (Tjong Kim Sang and Buchholz, 2000) dataset is imported. CoNLL2000 contains the following tuples: word, POS, IOB label. Although, IOB labels can also be tuned successfully (as mentioned in section 2.), in this article feature tuning is limited to POS fine-tuning only. This strong limitation is only for simplifying and demonstrating the impact of the toolkit.

Moreover, there are plenty of methods for feature selection. These methods can handle many types of feature, and they reduce effectively their number. But automatic feature selection is able to distinguish between useful and unuseful features, and this process is a black box for a human. (In practice: it might show which feature columns can be skipped.) On the one hand, in our case we investigate only one feature (POS). On the other hand, this toolkit would like to keep the feature tuning supervised and controlled by a human. Therefore automatic feature selection was not applied.

The tuples of the dataset are simple, and the tool can demonstrate the basic idea on it: statistical suggestions help the linguist to make useful modification on the dataset. In practice, the tool is able to tune other features as well.

To sum it up, this tool can optimize one feature at a time, in our case POS is tuned. But it can be applied to any features (not only for POS), even to output of another feature selection methods. The tool was tested with English and Hungarian datasets.

Statistical or rule based chunkers perform better with features which occur always with the same IOB label. On this assumption, we would like to have such features. It can be achieved when features have low cardinality, and if more specific sub-features would cause higher kurtosis in the distribution of features and their IOB labels. In other words, if the number of IOB labels assigned to a given feature would decrease with a more specific feature, it should be used. For instance, supposing *NOUN* have 3 IOB labels (B, I, E) with the same probability, but a special subset *MISTER* has most of the time only label B, then usage of this sub-feature would help the system performance. This conditions are fulfilled at the case of POS not only in English but in Hungarian, too. In this article English POS is tuned, but similar features exist in agglutinative languages as well. For clarity, these new sub-features (subPOS) are used as new decision classes instead of the original features.

The toolkit investigates the training set, and makes suggestions for features classified to more than one different IOB labels. Of course, this can be normal (a noun can stand in different positions of a NP), but it can also indicate that certain part-of-speech (POS) categories should be split into more detailed POS tags for better IOB labelling. For instance, a word might behave differently in a given POS category than the others (with respect to labelling), therefore this word should get a new category. These suggestions are important, and the main function of the toolkit is to hunt for them.

The features of the toolkit:

- the best feature patterns are detected,
- ambiguous patterns are shown,
- features can be browsed with respect to IOB labelling,

[1] https://github.com/endredy/onlineChunkerToolkit

- new features can be defined, which are applied to the training/test sets,
- it may speed up feature tuning
- a regular-expression-based grammar is built automatically to verify quality,
- grammar rules can be added manually as well,
- it estimates an approximately F-score of the training set on the fly
- each suggestion comes from the training set only, the test set is kept separately. The test set is used only when final modifications are exported.
- modified training/test datasets can be exported
- the tool is open source

6. How it works

Firstly, toolkit investigates the training set, secondly, it offers feature suggestions in more ways: browsing POS tags (ordered by frequency/assigned IOB label number/usefulness respect to IOB labelling), listing all valid NP sequences with POS, and listing POS suggestions which might correlate better with IOB labels (generated from WordNet, details in Section 7.). If user accepts some suggestions (or create a new one), the dataset is automatically investigated again with the modified POS, and it starts over again. If a feature sequence (in our case POS sequence) always gets the same IOB label in the training set, then this pattern will be signed with a green check icon, and it will be put into a grammar set. Other patterns will be signed with a red x, more than one IOB labels are assigned to them. (Figure 2) However, accepting every green case would result in overfitting and in low recall. It is therefore important that all decisions are done by the linguist, the tool only prepares and suggests. If (s)he accepts general cases (not data specific ones), overfitting can be avoided.

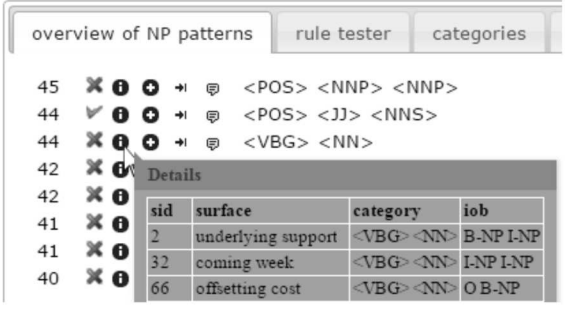

Figure 2: Red and green NP patterns with respect to IOB labels

At his moment, rules are used only for verifying F-score, they do not play role in the final exported dataset. The grammar contains regular expression rules, just like the grammar found in the Natural Language Toolkit (NLTK) (Bird et al., 2009). The difference is that our rules are executed from javascript in a browser, not in a standalone Python program, and the results can be seen on the fly in a window. Their syntax is similar, but the grammar engine is written from scratch in our case, serving two aims. First, it can give a fast feedback about the last modifications: whether they moved the dataset into a better state or not. Second, grammar rules can be tuned. Red patterns can be overviewed, and if a pattern seems to be acceptable, it can be put into the grammar with one click.

In addition, features can be overviewed with respect to IOB labelling: each feature shows its assigned IOB labels (from the training set). If a feature has only one label, it is the best case. (Statistical NP chunkers will learn it easily.) If not, then one can browse all its occurrences in the training set (Figure 3).

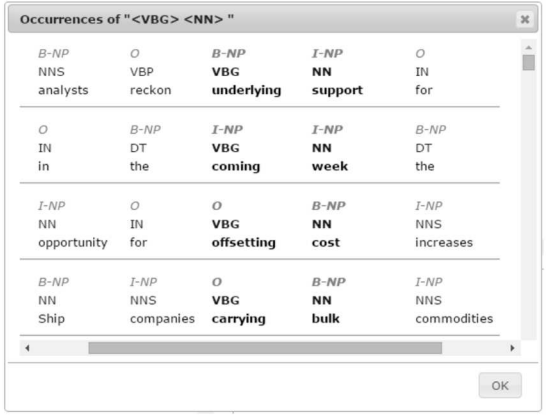

Figure 3: Browsing all occurrences of a POS pattern and its context with respect to their IOB labels

At this point, there are usually some words which behave differently than most words in the same category. For instance, in the CoNLL2000 dataset the word ”Mr.” has the part-of-speech tag NPP, but it behaves differently from the other proper names: ”Mr.” likes to stand at the beginning of a NP. (In other words it gets most of the time labelled as ”B”.) The toolkit supports to find easily that type of features, which get mostly one label (80%), and presents them to the user who can split the POS category by one click. Our toolkit gives the opportunity to create a new so called macro, and the given word will have the new feature instead of the original one. In this case ”Mr.” will get the tag MISTER, therefore it can be handled separately. This way, the feature tag of the odd word is replaced with a new one. The macro can be defined based on features, the surface form, the stem of the word or a regular expression pattern as well (latter two options are for future usage).

When features are browsed, one can define a new macro with one click. The program automatically writes a macro and gives an opportunity to modify it.

Macro definition supports the creation of a more powerful grammar: not only features (for example part-of-speech tags) but the surface form of words can also be added to the rules by their macro name. (Figure 4)

This is made automatically by the toolkit: if new macros are applied (a button is pushed), then the training set is converted with macros, and the extraction of the best

Figure 4: Examples for defining new categories (macros): based on surface form of word, stem or regex

feature sequences is done with these new features. If the F-score is not increased, then the added macros are useless. They should be dropped and other ones can be added. The evaluation takes only a few seconds, much faster than, for instance, tests in real statistical NP chunkers. The F-score is an approximately value based on a simple NP grammar, which is built automatically from features and IOB labels. It is not a real F-score, just a metric how features correlate to IOB labels. It is counted on training set, and it might show how "IOB friendly" the dataset is. (Test set is separated, it is used only at exporting modified dataset.)

7. WordNet helps discovering new features

Another source of possible feature suggestions is WordNet synsets (Miller, 1995). First of all, synsets and hyponyms of each word were generated into a dataset (separated by slash). Second, these synsets were split by slash, and every single synset was investigated with respect to IOB labelling.

category	sum	label	freq	%
period.n.07	207	I-NP	197	95.17%
		B-NP	10	4.83%
side.n.10	84	I-NP	80	95.24%
		B-NP	4	4.76%
large_indefinite_quantity.n.01	1654	I-NP	1577	95.34%
		B-NP	77	4.66%
about.s.01	258	B-NP	246	95.35%
		I-NP	12	4.65%

Figure 5: Examples for feature suggestions generated from WordNet

Finally, synsets were sorted by the number of IOB labels and frequencies: best synsets have one or rarely two labels. If the user likes any of the suggestions, it can be added by a single click. WordNet synset suggestions will be used instead of the originally POS in the exported dataset, and every related word will have it. For instance, POS of the word *period* could be *period.n.07* instead of *NN*, because the former correlates better to its IOB labels than the latter one. Some suggestions are shown in Figure 5.

8. Experimental results

The CoNLL2000 dataset was imported into the toolkit. As we focused on NP chunking, other labels (VP, PP, etc.) were eliminated from the dataset and they were changed to the label "O".

During the evaluation, first, POS features were evaluated by their co-occurrences with IOB labels. Second, some macros were added to the dataset, concentrating on words having the same POS category, but different behaviour. The best WordNet suggestions were also added, and finally, training and test data were exported and used in several NP chunkers.

All these steps were made with the help of the toolkit, and only POS were fine-tuned. Results were measured on this modified CoNLL2000 dataset.

The baseline test was made with the unigram and bigram NP chunkers of the NLTK toolkit. In addition, a new statistical tagger was also used in the evaluation: HunTag3 (not yet published), which is a general-purpose sequential tagger with linear SVM classifier (Maximum Entropy) of Liblinear (Fan et al., 2008) and Maximum Entropy Markov Models (Ratnaparkhi et al., 1996). This tool is developed in an ongoing parallel project.

method	**F-score**
NLTK - unigram chunker	
with original tags	83.2%
with modified tags by toolkit	**83.8%**
NLTK - bigram chunker	
with original tags	84.5%
with modified tags by toolkit	**86.1%**
HunTag3	
with original tags	92.68%
with modified tags by toolkit	**92.74%**
voting system between more chunkers (Shen and Sarkar, 2005)	
with original tags	92.74%
with modified tags by toolkit	**94.12%**
voting system between more chunkers + HunTag3	
with original tags	93.13%
with modified tags by toolkit	**94.59%**

Table 1: CoNLL2000 test runs with and without the toolkit, only POS data were modified

Results show the usefulness of the toolkit: it could help every NP chunker to reach higher F-scores (Table 1).
The current state-of-the-art NP tagger is SS05 (Shen and Sarkar, 2005), which achieves 95.23% on the CoNLL2000 dataset. Its method is based on voting between more data representations, which means different IOB labelsets (IOB1, IOB2, IOE1, IOE2, O+C) and each IOB label is completed with POS. This solution modifies the IOB labels, specialized them with POS. This idea is similar to our approach, however, we try to fine tune the POS info instead of the labels. SS05 voting system was reimplemented in python: conversions between different IOB labelsets, adding POS to labels, training each representation with TnT tagger of NLTK (Bird et al., 2009), converting results to a common labelset, and voting between the results. HunTag3 was also added as a 6th system (see Table 2), and

it could improve the final F-score of the voting (+1.4%). Even though our results are lower than SS05, but our aim was to demostrate the power of the toolkit, when using to boost the results of existing chunkers.

voting format	with original POS	modified POS by the toolkit
IOB1	92.01%	93.57%
IOB2	90.71%	92.04%
IOE1	90.64%	92.18%
IOE2	88.67%	89.96%
O+C	90.52%	91.71%
after voting	92.74%	**94.12%**
after voting, HunTag3 added	93.13%	**94.59%**

Table 2: Detailed results of the voting system between different data representations: not only IOB labels (based on SS05) but POS were also modified by the toolkit. It could improve each voting format.

9. Conclusion

Hundreds of features increase not even the training space and time, but contradictory features may prevent to reach higher F-score in the task of NP chunking. Manual detection of missing or inaccurate features is more than problematic. However, the tool presented in this paper could easily improve the quality of the features quickly (+2% F-score improvement).

No doubt, the results of this toolkit could be reached with command line tools as well. However, the steps of the data tuning were made in a user friendly web application, with mouse clicks in a short time, without any advanced IT user skills. This toolkit is designed for linguists who are not developers but have good intuitions and a web browser.

This toolkit provides an automated way of training set analysis. The user is guided through the problematic cases by the program. A person cannot overview all the specific details of a corpus, while a machine can not make abstraction. This toolkit connects the two approaches: details are shown to the user who can decide on the use of the features.

Any learning algorithm can work better with a more consistent dataset and with more IOB-friendly features.

10. Future plans

The toolkit can be used to build a regular expression based NP grammar. At this moment it is used only for the verification of feature tuning. It would be interesting to build a simple rule-based NP chunker with this grammar. Rules may contain POS categories, macros, surface forms of words with the power of regular expressions. An algorithm is needed which merges rules in the simplest form and patterns should be imported from a bigger corpus (e.g. web). Then, it may result in a new NP chunker. Of course a language can not be described with finite patterns, but if the corpora is huge enough, this idea may work.

Acknowledgments.

I would like to show my gratitude to Dr Gábor Prószéky for motivation, and I thank Dr Nóra Wenszky, Borbála Siklósi and the anonym reviewers for their comments.

11. References

Bird, Steven, Ewan Klein, and Edward Loper, 2009. *Natural language processing with Python*. O'Reilly Media, Inc.

Déjean, Hervé, 2000. Learning syntactic structures with xml. In *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th CoNLL - Volume 7*, ConLL '00. Stroudsburg, PA, USA: ACL.

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Johansson, Christer, 2000. A context sensitive maximum likelihood approach to chunking. In *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th CoNLL - Volume 7*, CoNLL '00. Stroudsburg, PA, USA: ACL.

Koeling, Rob, 2000. Chunking with maximum entropy models. In *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th CoNLL - Volume 7*, CoNLL '00. Stroudsburg, PA, USA: ACL.

Miller, George A, 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Molina, Antonio and Ferran Pla, 2002. Shallow parsing using specialized hmms. *The Journal of Machine Learning Research*, 2:595–613.

Osborne, Miles, 2000. Shallow parsing as part-of-speech tagging. In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th CoNLL - Volume 7*, ConLL '00. Stroudsburg, PA, USA: ACL.

Ratnaparkhi, Adwait et al., 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1. Philadelphia, PA.

Shen, Hong and Anoop Sarkar, 2005. Voting between multiple data representations for text chunking. In Balázs Kégl and Guy Lapalme (eds.), *Advances in Artificial Intelligence, 18th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2005, Victoria, Canada, May 9-11, 2005, Proceedings*, volume 3501 of *Lecture Notes in Computer Science*. Springer.

Simon, Eszter, 2013. *Approaches to Hungarian Named Entity Recognition*. Ph.D. thesis, Budapest University of Technology and Economics Budapest.

Sun, Xu, Louis-Philippe Morency, Daisuke Okanohara, and Jun'ichi Tsujii, 2008. Modeling latent-dynamic in shallow parsing: a latent conditional model with improved inference. In *Proceedings of the 22nd COLING-Volume 1*. ACL.

Tjong Kim Sang, Erik F. and Sabine Buchholz, 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th CoNLL - Volume 7*, ConLL '00. Stroudsburg, PA, USA: ACL.