Statistical Noun Group Detector

Antonio CALZADA
Department of Computer Science
Lund University
dat95jca@hotmail.com

Abstract

Statistical noun group detectors and chunkers provide powerful means of detecting syntactically correlated non-overlapping parts of sentences.

This report describes discoveries made exploring statistical noun group detection based on Support Vector Machines (SVM) applied on data from the CoNLL-2000 shared task.

1 Introduction

Text chunking consists of dividing a text in syntactically correlated parts of words. For example, the sentence *He reckons the current account deficit will narrow to only # 1.8 billion in September* . can be divided as follows:

[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September] .

The shared tasks of CoNLL provide excellent reference material and results.

In the shared task of CoNLL-2000 full phrase part chunking is explored.

In shared task of CoNLL-1999 NP bracketing is explored. This consists in identification of all noun-phrase structures allowing multiple levelled groups where a for example a noun-phrase may be identified as being decomposable into smaller noun-phrases.

Noun group detection. Also known as noun phrase (NP) chunking is a simple and robust alternative to full parsing for segmenting a text into syntactically correlated parts.

While this report specifically explores detection of noun groups, many times the same methods can be applied to other group detection problems like full phrase part detection and identification of names of companies and people in texts.

Because statistical methods and learning algorithms are used instead of for example hand made rules, the implementation can easily be adapted to different languages and types of text.

2 Segmentation and labelling

Segmentation and labelling are two of the most common operations in natural language processing. These two operations are strongly related. While segmentation divides a stream of characters into linguistically meaningful pieces, labelling classifies those pieces.

There are many different ways a text can be segmented, most notably: bulletins, pages, sections, sentences, phrase parts, words and word stems.

Segmentation might be done at more than one level. When classifying news bulletins it might suffice to have two levels of segmentation. First the text would be segmented into bulletins and then into sub-segments of keywords and non-keywords. This would contrast full parsing, where text is segmented into hierarchical structures of unlimited depth.

Labelling is characterized by the set of labels used and their meaning. It may range from sentence identification by enumeration to tagging using an extensive set of part-of-speech (POS) tags.

3 SVM

Support Vector Machines (SVMs) are used for solving classification and regression problems, very much like neural networks.

They are trained using data sets of attributes (features) and corresponding target value (class label). A trained SVM model can then be fed sets of features that it will attempt to classify correctly.

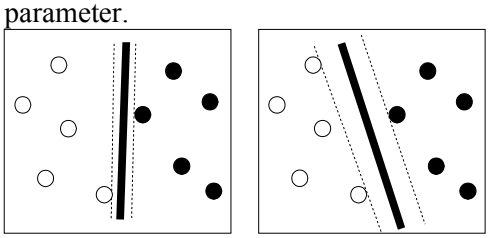
The SVM model training works by mapping the training vectors into a higher dimensional space. During training the SVM engine attempts to maximize the margin between critical values and a separating hyperplane. See (Drawing 1) for an illustration where the dark line represents a projection of the hyperplane dividing the dataset. The thinner dotted lines mark the distance between the plane and the closest data point.

There are a number of configuration parameters that can be tuned for the task at hand, the most common are kernel function, γ and C.

The heart of the SVM engine is a pluggable kernel function controlling the creation of the hyperplane. Some examples are: linear, polynomial, radial basis function (RBF), and sigmoid.

Depending on the type of the function, a number of configuration parameters may be available. The γ parameter is common to most kernel functions.

Since it may not be possible to place the plane so it correctly classifies the training data it is allowed to incorrectly classify training values but at the cost of a penalty that is to be minimized. The severity of this penalty is controlled by the C parameter.



Drawing 1 SVM Classification optimization

4 NP identification using SVM

The basic steps for applying SVM to NP detection are:

- Selecting appropiate features.
- During training:
  - Scale and encode features for train data.
  - Select kernel function and trim parameter.
  - Train SVM model.
- During testing:
  - Scale and encode features for test data.
  - Let SVM model classify test data.
  - Decode classification labels.

Since SVM works with points in a mathematical space the words and tags in the natural language material needs to be encoded into numbers.

SVM like most learning algorithms thrive on information. But it's important that both the training data and the encoding is not biased fooling SVM into identifying patterns that are not applicable to the test data.

5 Test, training and evaluation

The material I used is the same as used at CoNLL-2000 and in turn originally produced by (Ramshaw and Mitchell, 1995).

The corpora contain one word per line and each line contains six fields of which the first three fields are relevant: the word, the part-of-speech tag assigned by the Brill tagger and the correct IOB tag showing phrase par limits.

Words can be inside a NP (I) or outside a NP (O). In the case that one NP immediately follows another NP, the first word in the second base NP receives tag B.

The source corpora of the data is sections of the Wall Street Journal (WSJ), part of the Penn Treebank (Marcus et al., 1993). Sections 15-18, 211727 tokens are used as training material and section 20, 47377 tokens as test material.

Three values are used to measure performance:

- precision, percentage of detected noun phrases that are correct ,
- recall, percentage of noun phrases in the data that were found by the classifier,
- and F-beta, provides a collected measurement of the previous two values evaluated as (2*precision*recall)/ (precision+recall)

Results are measured up against a baseline value provided by a simple unigram tagger (tagging IOB tags instead of POS tags).

6 The Demo Program

The purpose of the demo program is to demonstrate how an arbitrary text provided the user is tagged and NP chunked by the implemented methods.

The demo is implemented as an interactive console program. Once it is started an introductory message and a prompt is shown. The prompt accepts the commands described in (Table 1).

The tagger used is a simple back-off tagger composed of in order of preference: a trigram tagger, a bigram tagger, a unigram tagger, and as a last result defaulting to the NN tag.

Table 1Demo program commands

command	Description
chunk	Chunks the provided text. Tagger and SVM engine has to be loaded.
create_svm [word count]	Creates a SVM engine and trains it using CoNLL-2000 train data. A word count can be provided to limit the size of the corpora used.
create_tagger [word count]	Creates a tagger and populates it with data from the CoNLL-2000 train data. A word count can be provided to limit the size of the corpora used.
evaluate_tagger	Evaluates the currently loaded tagger.
exit	Exits the program.

help [command]	Shows either available commands or information about a command if provided.
history	Provides a list of executed commands.
load_svm	Loads a svm model from provided filename or svm.pkl if none given.
load_tagger	Loads a tagger from provided filename or tagger.pkl if none given.
save_svm [filename]	Saves a svm model to provided filename or svm.pkl if none given.
save_tagger [filename]	Saves a tagger to provided filename or tagger.pkl if none given.
shell [command]	Executes the provided command in a spawned shell.
tag text	Tags the provided text using the loaded tagger. Also saves the output to the file tagged.txt.
test	Creates a tagger and a SVM engine and then tags and chunks a test text.
! [command]	Same as the shell command.

7 Results

Building on the baseline (F-beta = 79.99) implementation by using n-tagger showed showed some improvements that quickly tapered off with tagger complexity.

Using SVM on half the training set with C=64 and γ=64 produced F-beta=80.99 when not using context based features. When adding the previous context based POS tag to the features F-beta improved to 86.60.

8 Conclusion

Although my results are not that great, they show that just adding some context data shows a good improvement. Providing more an better features would give very good results.

9 References

Dimitrios Kokkinakis and Sofie Johansson Kokkinakis. 1999. *A Cascaded Finite-State Parser for Syntactic Analysis of Swedish*, In: "Proceedings of EACL'99", Bergen, Norway.

Taku Kudo and Yuji Matsumoto.2001. *Chunking with Support Vector Machines*, In: "Proceedings of NAACL 2001", Pittsburgh, PA, USA.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz. 19993. *Building a large annotated corpus of English: the Penn Treebank*, Computational Linguistics.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. *Text Chunking Using Transformation-Based Learning*. In: "Proceedings of the Third ACL Workshop on Very Large Corpora", Association for Computational Linguistics.