GslcTagger for parts-of-speech tagging of transcribed speech: A technical report (draft)

Abdul Rahim Nizamani

SCCIIL Center for Interdisciplinary Research,
Department of Applied Information Technology,
University of Gothenburg, Sweden
`abdulrahim.nizamani@chalmers.se`

13 November 2013

Contents

Summary

This report presents the design and technical details of the “GslcTagger”, a trainable parts-of-speech tagger for transcribed speech. This POS tagger was developed to tag the spoken Swedish language corpus called the GSLC (Göteborg Spoken Language Corpus). It is a rule-based tagger based on the Brill’s rule-based tagger design (Brill, 1992), and is implemented in Haskell. Design of the model and adaptations for spoken language are described, and relevant algorithms are listed in the appendix. File syntaxes and commands are described and illustrated with examples.

Chapter 1

Introduction

Computational parts-of-speech (POS) tagging was initially developed using manually engineered rules (Klein and Simmons, 1963; Harris, 1962). As larger corpora became available and computing power increased, automatically trainable taggers were developed. POS taggers today generally fall in two categories: stochastic and rule-based.

Rule-based training of POS taggers was described by Brill (1992). Such systems can achieve similar performance to stochastic taggers with a small number of automatically induced non-stochastic rules.

Most POS taggers today are highly specialized to achieve high accuracies, up to 97% without human intervention. The current best score for automatic POS tagging is about 97.3%, and suggestions to reach close to 100% accuracy are discussed by Manning (2011). However, almost all of the current tagging systems have been geared towards corpora of written language, with little effort made for transcribed speech and conversations.

Nivre and Grönqvist (2001) have described a method for training a POS tagger to tag spoken language corpus. Their method uses written language corpus to train a stochastic POS tagger, and then adapts it for tagging a transcribed spoken corpus. They have reported an accuracy of 95% for a tagset with 23 tags, and 97% for a smaller tagset with 11 tags (listed in appendix A).

GSLC (Göteborg Spoken Language Corpus) is a corpus of spoken Swedish, constructed from different scenarios, and has been described by Allwood et al. (2002). Currently it contains more than 1.45 million words from 28 different activities. Transcriptions are coded in the Göteborg Transcription Standard (GTS) that is described by Nivre et al. (1999).

GSLC was first tagged with parts of speech by Nivre and Grönqvist (2001). However, lately it has grown considerably with the inclusion of new transcriptions. To facilitate research and exploration of the corpus, a web-based online corpus browser called Brusaren is developed and is available freely to the subscribers. Recently, the new corpus has been re-tagged with POS, and simple statistics such as word frequencies for a given tag are now available in the Brusaren explorer.

To achieve this, a new rule-based trainable POS tagger has been designed to facilitate semi-automatic tagging of the data, in order to reach accuracy close to 100%. The new tagger is named GslcTagger. It is written in Haskell and is based on the design of Brill (1992). The current model is an adaptation of the original design and focuses on tagging speech transcriptions. Being rule-based and semi-automatic, little human effort can easily increase the accuracy of tagging by manually identifying errors and adding appropriate rules for correction. Although the current accuracy is 97%, same as previous, the new design helps correct common errors by manually introducing new rules and retagging the data.

GslcTagger is designed for the GSLC corpus that has been coded with GTS. However, the tagger does not operate on the GTS coded text directly, rather works on a plain-text version of transcriptions. This enables the tagger to be used for any other speech transcriptions that can be converted to plain text files with each utterance occupying a separate line in the text.

This document describes technical aspects of the GslcTagger. Chapter 2 presents the design of the tagger and an overview of its algorithms. Chapter 3 gives the technical details and syntax of the files produced/used by the tagger. Chapter 4 is a reference of the tagger commands to perform different tasks. In the end, a summary of results is provided in chapter 5.

Chapter 2

Design

The GslcTagger is designed as a rule-based, automatically trained POS tagger based on the model described by Brill (1994).

2.1 Design

The general design of the tagger follows a pattern similar to the one described by Brill (1994). Untagged text first goes through an initial tagging process that does not consider contextual rules. After initial tagging, the tagged text goes through a second process of applying contextual rules, which correct errors by applying transformations learned earlier (in the training phase).

Initial tagging of the data is achieved by assigning each word its most likely tag from the *count* file, which contains the count of each word/tag pair (*count* files are described in section 3.2). Here, *most likely* refers to the most frequent tag for a word. Unknown words are tagged as *noun*, since nouns are the most frequent category in the written corpora. An overview of the initial tagging process is given in Algorithm 1 (appendix B).

After the initial tagging, the tagged text is then sent to the contextual model which applies rules (originally called transformations by Brill) to correct errors. Rules are applied in the same order in which were generated or learned. Each rule transforms the tagged corpus into a new version that becomes the input to the next rule.

2.2 Training

Training of the tagger is carried out in two steps. The first step produces a *count* file that contains summary of the word/tag pairs. This file is used in the initial tagging process.

Another *count* file is created for foreign words, from any language which has been used in the spoken corpus apart from the main language. For GSLC, the second most used language is English, and the *count* file for foreign words was constructed from the Brown corpus (Francis and Kucera, 1979).

The second training step learns contextual rules from a smaller pre-tagged corpus of spoken language. For better results, first a larger written corpus is used for learning the contextual rules for written language. Then a smaller spoken corpus is used for learning further rules, which build on the previously learned rules.

Rules are found by training the tagger on a tagged corpus, using the following templates:

For every word in an utterance, change tag **x** to tag **y** when

1. The current word is w.
2. The current word is tagged t.
3. The preceding (or following) word is w.
4. The preceding (or following) word is tagged t.
5. Any of the preceding (or following) n words is tagged t ($2 \leq n \leq 3$).
6. All of the preceding (or following) n words are tagged $t_1, \ldots, t_n$ ($2 \leq n \leq 3$).
7. The preceding and following tags are t_p and t_f.
8. The current word is w and the preceding (or following) word is w'.
9. The current word is w and the preceding (or following) word is tagged t.

Algorithm 2 in appendix B lists the steps used by the tagger to learn new contextual rules.

Brill describes templates for linguistic transformations to increase accuracy of tagging unknown words. These rules basically find patterns in word spellings and are not language-specific. Though highly desirable, this part of his model was omitted in the current design due to following reasons:

- Transcriptions of speech contain many words with nonstandard spellings, thus may not be ideal for learning linguistic rules for unknown words.
- Speech corpora are usually small. Accuracy on unknown words can easily be boosted by manually providing best tags for frequent unknown words.

When the tagger is used to tag a plain text file, it produces a list of unknown words sorted by their frequency of occurrence. A simple way to increase tagging accuracy on unknown words is to manually tag the most frequent unknown words from this list, and then merge the result to the count file. In the next run, there will be much less unknown words tagged incorrectly.

2.3 Implementation

The tagger is implemented in Haskell, a high-level purely functional programming language. Haskell is somewhat slower than other imperative programming languages, however it saves programming effort rather than computational time. Current computational architecture is sufficiently powerful, but the human effort required in developing a software is rather expensive. Thus, the implementation took considerably less time than it would have if written in imperative languages.

Chapter 3

Syntax of the files

3.1 Corpus file

The corpus files used in the GslcTagger are plain text files encoded with `utf8` encoding. Each line contains a single sentence/utterance with words tagged as "word/tag", i.e., each word is followed by a forward slash "/" and a tag. Words can contain a slash, as only the last slash character is used to separate a word from its tag. For example, the token *a/b/noun* is parsed as the word *a/b* tagged as *noun*.

A corpus file is coded using standard orthography, similar to written language except that there are no punctuations and capital letters. GTS transcriptions in their source format are not suitable for use in the tagger, and need to be cleansed and formatted properly. This is easily done by downloading a formatted text file from the Brusaren2 corpus browser, whose admin page contains a button for it. The text is retrieved from the standard orthography column of the `utterances` table of database, which is already pre-formatted. An excerpt from it is given below as an illustration.

```
ja det stämmer
ja
de unga veteranerna
de unga veteranerna
ja:
```

3.2 *Count* file

A "*count*" file contains the counts (statistics) of all word/tag pairs from a pre-tagged corpus. Usually, large pre-tagged corpora of written language can easily be acquired, which can be used to build the *count* file for GslcTagger. This file is used for initial tagging of the untagged data of the spoken corpus.

The syntax of data in the *count* file is rather simple. All word/tag pairs are listed in a line followed by their total count, separated by a space.

```
word tag count
```

The *count* file needs to be sorted by words, however the tags can appear in any order. It is encoded with the `utf8` encoding. An example is following, taken from the GP (Göteborgs Posten) corpus:

```
förste num 5983
förste verb 2
förste adj 14
förste noun 2
```

3.3 Rules file

Rules are specified with the following format:

```
CURRENT TAG -> NEW TAG :: CONTEXT [&& CONTEXT ...]
```

where the context can be any of the following:

```
One (position) tag
OneW (position) word
Both previoustag nexttag
BothW currentword (position) word
BothT currentword (position) tag
Any (position) [list of tags]
All (position) [list of tags]
```

For example, consider the following rule:

```
"verb" -> "adj" :: OneW (-1) ett
```

It changes the tag of current word from “verb” to “adj” whenever the previous word (i.e., word at position -1 relative to the current word) is *ett*. Note that the current tag can instead be replaced with a wildcard (underscore “_ ”) to match with any tag. The rule then will be written as:

```
 _ -> "adj" :: OneW (-1) ett
```

More than one contexts can be included in a single rule, in which case all contexts will be matched before replacing the tag. Contexts are joined by the symbol &&. An example is following:

```
"noun" -> "verb" :: OneW (0) höller && OneW (1) på
```

This rule will change the tag of the word *höller* (current word, indicated by positon 0) from “noun” to “verb” if the next word (indicated by position 1) is *på*.

Chapter 4

Manual of commands

The GslcTagger performs a number of functions, specified with a *command*. Some commands are necessary to build the training data and use it for tagging, others perform a number of optional tasks. Commands are specified with the following general syntax:

```
GslcTagger command [options]
```

A summary of all commands and their options can be printed by using the option `-h` or `--help`:

```
GslcTagger -h
```

4.1 Building a *count* file

A *count* file can be built from any pre-tagged corpus. The `count` command is used to build a *count* file from a corpus, using the following syntax:

```
GslcTagger count corpusfile outputfile
```

where,

`corpusfile` =	name of the text file containing tagged data. This file must be in `utf8` (8-bit unicode) encoding.
`outputfile` =	filename for storing the count result (a `.count` extension can be used)

Once a *count* file is built, it can be updated from any other tagged data (e.g., another pre-tagged corpus) using the `update` command. Following syntax is used:

```
GslcTagger update countfile corpusfile
```

A *count* file can also be updated using another *count* file, in which case the second is merged into the first. The `merge` command is used with the following syntax for merging *count* file B into *count* file A (only file A is modified, file B remains unchanged).

```
GslcTagger merge countfileA countfileB
```

Note that the *count* file B does not need to be sorted in any order.

The `merge` command is also useful for building a new *count* file from only statistical data (when the full tagged corpus is not available and only word/tag statistics are provided, such as the GP corpus). To do so, first a blank *count* file is created to store the results, and then data files containing statistical information are merged into it one by one.

4.2 Training and running the tagger

After building a *count* file from a large tagged data, the tagger is trained on a (small) tagged corpus of spoken language, from where it learns the contextual rules for the speech corpus. The rules are stored in the `rulefile`, which is updated by the `train` command. If there are any existing rules in this file, the tagger uses them in the initial tagging process, before continuing to learn new rules. Thus the training can be interrupted at any time and then restarted without any side effects.

The `run` command is used to tag any untagged data contained in a plain text file. Its options are similar to those of the `train` command. Syntax for these commands is listed below:

```
GslcTagger train -i inputfile -r rulefile [options]

GslcTagger run -i inputfile [options]
```

where the options are listed below,

`-i inputfile (train)` = the tagged speech corpus used for learning contextual rules

`-i inputfile (run)` = the untagged plain text file that is to be tagged

`-c countfile` = the *count* file

`-f foreigncount` = the *count* file for foreign language words

`-r rulefile` = the file containing contextual rules

`-csv` = indicates that the input file is in `csv` format (only used in `run` command)

These options can be provided to the program in any order. The options listed outside the square brackets in the syntax are necessary, whereas others are optional.

4.3 Utilities

4.3.1 Comparing results

The `compare` command can be used to compare two tagged versions of a text. It can be used for instance to compute tagging accuracy of the tagger, by first tagging a test file and then comparing the result with the source. Its parameters are two tagged files (with no particular order) and optionally a *count* file.

```
GslcTagger compare file1 file2 [countfile]
```

If the *count* file is specified, it produces separate results for known and unknown words.

4.3.2 *Count* file statistics

The `stats` command is used to compute a summary of the statistics in a *count* file. It will show the total number of words and types, and will also print the number of words for each tag.

```
GslcTagger stats countfile
```

4.3.3 Word frequencies

Word frequencies for a particular tag in a *count* file can be produced using the `freq` command. Results are stored in the `outputfile`.

```
GslcTagger freq tag countfile outputfile
```

where,

`tag` = the tag for which word frequencies are to be computed
`countfile` = the *count* file
`outputfile` = filename to store word frequencies

Chapter 5

Results

The GSLC corpus at its present state comprises some 1.45 million words. GslcTagger was used to tag the words for their part-of-speech word class with automatically learned rules. Further rules were added to correct manually identified errors, and this process is still on-going. The accuracy of the tagger was 97% before introducing the manual rules and 98% thereafter, thus it can be estimated that the current error rate is less than 2% in the full corpus.

The corpus browser was updated to show the POS statistics and example utterances for each class and word. It now includes the following listings:

1. Division of word classes in the corpus
2. Frequencies of words for a particular tag
3. Utterances containing a word tagged as a particular tag (words can be chosen by applying regular expressions)

The above listings can be generated for the complete corpus as well as for filtered results. All the regular filters of the corpus browser are available in the POS section, such as activity type, gender of the speaker, or number of participants in the transcription.

Table 5.1 presents the division of the corpus in word classes. The *previous* column shows their relative occurence in a previous version of the corpus, listed by Allwood (2000).

	tag	percentage	previous
1	pron	22.26%	22.55%
2	verb	19.42%	19.37%
3	adv	16.25%	16.32%
4	noun	12.44%	12.22%
5	conj	8.38%	8.40%
6	prep	7.05%	7.13%
7	fb	5.65%	5.80%
8	adj	4.36%	4.11%
9	ocm	2.30%	2.26%
10	num	1.62%	1.55%
11	int	0.25%	0.28%

Table 5.1: Division of word classes in the GSLC corpus

Chapter 6

Conclusion

The GslcTagger is a rule-based tagger and it has achieved similar performance on the GSLC corpus as the previous stochastic tagger. Given this, it has certain advantages over a stochastic tagger.

First and foremost, it allows for manual addition of new rules for correcting common errors, thus further increasing the tagging accuracy with human effort. This also enables modifying the tags for a word by introducing a rule, as the word classes often overlap and different researches can categorize words differently.

Secondly, the rules are written in a plain text file and are fully transparent to a reader or a researcher, unlike stochastic tagger which operates in non-transparent mode.

Bibliography

Jens Allwood. Talspråksfrekvenser: Ny och utvidgad upplaga. *Gothenburg Papers in Theoretical Linguistics*, S 21, October 2000.

Jens Allwood, Leif Grönqvist, and Elisabeth Ahlsén. Göteborgkorpusen för talspråk. *Nydanske Studier, Akademisk Förlag: Copenhagen*, 2002.

Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the workshop on Speech and Natural Language*, pages 112–116. Association for Computational Linguistics, 1992.

Eric Brill. Some advances in transformation-based part of speech tagging. *arXiv preprint cmp-lg/9406010*, 1994.

W Nelson Francis and Henry Kucera. Brown corpus manual. *Brown University, Department of Linguistics*, 1979.

Zellig Harris. String analysis of language structure. *Mouton and Co., The Hague*, 1962.

Sheldon Klein and Robert F Simmons. A computational approach to grammatical coding of english words. *Journal of the ACM (JACM)*, 10(3): 334–347, 1963.

Christopher D Manning. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*, pages 171–189. Springer, 2011.

Joakim Nivre and Leif Grönqvist. Tagging a corpus of spoken swedish. *International Journal of Corpus Linguistics*, 6(1):47–78, 2001.

Joakim Nivre, Jens Allwood, Leif Grönqvist, Magnus Gunnarsson, Elisabeth Ahlsén, Hans Vappula, Johan Hagman, Staffan Larsson, Sylvana Sofkova, and Cajsa Ottesjö. Göteborg transcription standard. *Version*, 6:38, 1999.

Appendix A

Tagset used to tag the GSLC corpus

	tag	description	corresponding small tags
1	adj	adjective	(jj,pc)
2	adv	adverb	(ab,ha,hs,pl)
3	fb	feedback word	
4	int	interjection	(in)
5	conj	conjunction	(kn,sn,ie)
6	noun	noun	(nn,pm)
7	num	numeral	(rg,ro)
8	ocm	own communication management	
9	pron	pronoun	(dt,hd,pn,ps,hp)
10	prep	preposition	(pp)
11	verb	verb	(vb)

Appendix B

Algorithms

```
Input: untagged corpus
Data: count file c
Data: foreign count file f
Result: tagged version of the corpus
foreach word w in corpus do
    if first or last character of w is '+' then
        tag w with "ocm"
    else if w can be parsed as numeral then
        tag w as "num"
    else if w is a member of c then
        let tag = the most frequent tag of w in c;
        if tag is neither "uo" nor "noun" then
            tag w with tag
        else if w is a member of f then
            tag w with its most frequent tag from f
        else
            tag w as "noun"
        end
    else if w is a member of f then
        tag w with its most frequent tag from f
    else
        tag w as "noun"
    end
end
```

Algorithm 1: GslcTagger initial-tagging algorithm

```
Data: initially tagged corpus ($corpus_i$)
Result: New contextual rules
repeat
    generate all rules from templates;
    forall the rules do
        tag $corpus_i$ with rule;
        netscore = corrections - new errors;
    end
    bestRule = rule with the highest score ($score \geq 1$);
    add bestRule to the rules file;
    $corpus_{i+1}$ = apply the bestRule to $corpus_i$;
    repeat with $corpus_{i+1}$;
until no rule has score $\geq 1$;
```

Algorithm 2: GslcTagger rule-learning algorithm