dTagger: A POS Tagger

Guy Divita, Allen C. Browne, Russell Loane
National Library of Medicine, Bethesda, Maryland

ABSTRACT

The Lexical Systems Group at the National Library of Medicine (NLM) has developed a Part-of-Speech (POS) tagger[1] to be freely distributed with the SPECIALIST NLP Tools[1]. dTagger is specifically designed for use with the SPECIALIST lexicon [2,3] but it can be used with an arbitrary tag set. It is capable of single or multi-word chunking. It is trainable with previously annotated text and in development is a version that is tunable with untagged text. The tagger allows users to add local lexicon content. It can report likelihoods for each sentence tagged. New words seen while tagging (the unknowns) are handled by shape identification including heuristics based on suffix statistics gleaned during the training. The performance of the supervised training is noted to be 95% on a modified version of the MedPost hand annotated Medline abstracts. Eight percent of the terms within this corpus were multi-word entities.

BACKGROUND

POS taggers resolve Part-of-Speech ambiguities when a lexical item such as *report* occurs in more than one part of speech. *Report* is both a noun and verb. POS taggers are often employed to aid in the task of determining phrase boundaries and thus the extraction of noun phrases. Noun phrase extraction is essential to indexing and retrieval within many tasks such as search engines, information extraction and categorization.

The SPECIALIST textTools [1], a Java based, open source suite of Natural Language Processing utilities, initially contained only rough heuristics in lieu of a POS Tagger. At NLM, we have used the Xerox Parc POS tagger [7] and recently MedPost/SKR, the Java implementation of the MedPost POS Tagger [4]. Although the MedPost/SKR POS tagger has performed well, it is not specifically tied to the SPECIALIST Lexicon, no trainer has yet been published, and the tagger tokenizes at a single and hyphenated word boundary, thus missing multi-word lexical elements (*LE*s) cataloged in the lexicon. The textTools employed additional machinery to identify LEs after tagging at the single word level.

Our motivation was to create a tagger that can be freely distributed with the textTools as an option that would allow users to train, customize, and, if necessary, modify it to suit a broad range of tasks.

PROJECT FEATURES

The next sections describe the major components of the tagging task. These include programs needed to train and use the tagger and the major modules within these programs.

A Hidden Markov Model

A Hidden Markov Model and Viterbi algorithm are used, similar in spirit to the algorithms described in Manning and Schütze[5]. Mathematical details underlying the model that dTagger is based upon can be found at [6]. A feature of this model is a normalization that produces a measure of the likelihood of the Viterbi solution (i.e., the best POS sequence). The normalization allows us to assign a significance metric to the tagged POS and compare the relative merit of nearby alternatives.

Lexical Lookup

dTagger is a component of the textTools; designed to work on its own, as well as to be employed within the NpParser tool. It contains our lexical lookup module. Lexical lookup is the task of segmenting the text into units that correspond to lexical entries from the lexicon. The important part of that task is to determine what possible parts of speech each unit can have. In prior versions of the textTools, lexical lookup and tagging were separate tasks. Hidden from view was the fact that a word based version of lexical lookup was again performed within the tagger, not necessarily to the same lexicon.

Many English lexical items are spelled with more than one orthographic word, for example "*Diabetes Mellitus*" or "*Myocardial infarction*". The SPECIALIST lexicon reflects this fact about natural

[1] http://SPECIALIST.nlm.nih.gov/dTagger

language and includes many orthographically multi-word items. The 2005 version of the SPECIALIST Lexicon contains 50.3% multi-word LEs. We believe that identifying the part of speech of these multi-word items directly can reduce the over all level of part of speech ambiguity. The ability of dTagger to deal with multi-word lexical items is a major innovation.

Since some applications may require that text be tokenized into single words, the textTools will maintain the option to analyze text into single or multi-word lexical items as well as provide an in-between capability of recognizing multi-word items only when they cannot be resolved into single word items. This option catches and correctly tags LE's such as "*in vitro*" with one tag, while analyzing *myocardial infarction* into two items with their own tags.

An arbitrary Tag Set

dTagger is designed to be used with the SPECIALIST lexicon but it is also intended to be tag neutral. Tags are enumerated in a file (tagset.txt).

The lexicon of possible tag assignments to be used by the tagger is represented in a set of pipe delimited UTF-8 relational files with the extension .lex. One of these .lex files is a case-sensitive index of lexical entries coupled with their parts of speech. Another file is lowercased for case insensitivity. And a third file is reserved for local content to provide users a way to add items locally to the lexicon. The tagger comes with tools to create .lex files from the SPECIALIST lexicon. Users who wish to use tags or tag assignments differing from those in the SPECIALIST lexicon can record their tag set in this local file and create a set of .lex files to act as the tagger's lexicon. This flexibility should also aid in using the tagger for languages other than English.

The Specialist textTools use dTagger configured with the SPECIALIST lexicon. It uses the 10 parts of speech (noun, adj, adv, etc) identified in the lexicon without inflection. These are also the categories needed by the textTool's noun phrase parser. Inflectional information present in the SPECIALIST lexicon can be discovered from the lexical entries but is not used in the part of speech tags.

dTagger will emit only the tags allowed in it's lexical (.lex) files. This creates a design challenge. Words that are lexically of one part of speech can be used in another. The participles of verbs for example are often used as adjectives. Gerunds are the present participles of verbs used as nouns. In configuring the dTagger for the textTools we have treated all present and past participles as potential adjectives and present participles as potential nouns. For example, in "growth factor *induced* gene", 'induced' is tagged as an adjective. For this purpose, a file (verbsAsAdjs.lex) has been created to represent these participles and gerunds.

Tagging Text

The tagger tool takes input text, and returns tagged text. Within the tagger API, the tag method returns an instance of Sentence, containing instances of LexicalElement, each having the tagger POS instantiated. A normalized likelihood is assigned to the sentence as a unique side effect. This likelihood can be thought of as a level of significance that can be compared against other sentences' likelihoods. This might turn out to be useful in those applications where you have two different interpretations of what the lexical elements are, say by means of shortest vs. longest spanning match techniques. When both are run through the tagger, the resulting likelihoods can be compared.

Training with Tagged Text

dTagger is distributed with a training tool that will train the tagger from a hand annotated corpus. The format for the hand annotated files are aligned two row entries, with the un-tokenized string as a third row:

```
PXXXXXXXXX|word 1|word 2|word 3|…
HandTagged|tag 1 |tag 2 |tag 3 |…
String    |word1 word2 word3 …
```

Table 1: Training Corpus Format

This format provides an easy way to edit the content. The output of the trainer is a series of files and indexes:

transitionProbs*N*.txt
emissionProbs*N*.txt
modelWords*N*.txt
dbx*N*/lexicalLookupIndexes
dbx*N*/llIdIndexes
newTaggedTrainerWords*N*.lex

Table 2: Training Output files

The following files are generated for debugging and fine tuning purposes:

lexicalLookupIndexes*N*.txt
corpusInconsistancies*N*.txt

Table 3: Training Debugging Files

The *N* in each of the filenames is incremented for version control.

There will be words within the annotated corpus that are not yet in the lexicon. As part of the training task, these new words are added, and reported as such within an additional *newTaggedTrainerWords.lex* file.

Handling Unknowns

The SPECIALIST Lexicon can never have full coverage of any growing corpus. From time to time, an unknown word will show up in text to be tagged. Within training, those words not found in the SPECIALIST Lexicon are added to the local lexicon (.lex) file. During tagging, when an unknown word is found, it is categorized, in part, by a morphology unit that guesses it's potential part of speech from suffix information and computed likelihoods of the POS's of words ending with that suffix[5]. During the training task, the last 10 characters from each word of the corpus and lexicon are taken and added to a reverse trie. The POS's are kept track of at each node. The trie is pruned, keeping only fruitful suffixes, along with the distribution of the POS's for each suffix. This forms the basis for the suffix based shape identification.

Handling Patterns

Certain patterns such as numbers should be caught before tokens are looked up in the index. The textTools has a number of pattern or shape recognizers, ranging from the identification of real numbers, percentages, fractions, to the identification of units of measure. The initial version of the tagger employs only punctuation, integer and real number shape identifiers before lexical lookup. In future releases, those shape identifiers such as the units of measure identifier (10 mg/k), levels of significance ($P < .005$), and sample counts ($N=20$) will be employed to more accurately identify those elements.

Additional Considerations

It was noted in an above section that an additional lexicon file was created to handle verbs acting as adjectives or nouns. It was empirically seen that these noun and adj forms were incorrectly being assigned to cases where they truly were verbs, but where the training had not gathered any statistics to correctly weight them. The tagger trainer was modified to include a penalty for those adj's and nouns derived from verbs, where no instance of these words were seen in the training corpus. Since this is a specific task based feature, it was made into an optional flag (--weightVerbsAsAdjs), turned on by default.

Training and Test Corpus

The training and test corpus from MedPost was used to train the tagger. The format was altered to a pipe delimited structure to allow for the identification of multi-word LEs. The MedPost training corpus contains hand annotations of 5716 sentences taken from Medline abstracts within the Genomics domain. In the original text, hyphenated words were considered one token as well as a few multi-word prepositions. Number ranges such as "1-5" were also considered one token. Otherwise, no other multi-word elements were identified. The SPECIALIST lexicon was not the sole source for the part of speech tags. Many function words were tagged with tags not annotated as such in the SPECIALIST Lexicon, as in the case of "both", assigned as a conjunction in all cases in the corpus, where as it is only considered a det or pron in the SPECIALIST Lexicon.

The original tags were transformed to those from the SPECIALIST tag set, multi-word LEs were identified and conflicting tag assignments were reported. Hyphenated forms that did not correspond to LE's in the lexicon were split apart into different tokens. A series of hand reviews and alterations were also performed by the 1st author to catch what the programs did not, and fix what the program broke. During the training, it was noted that this corpus contained a total of 151,043 LE's, 139,015 of which were single token LE's, and 12,028 (8%) were multi-token LEs. There were 33,149 tokens within the set of multi-token LEs. Within this corpus there were on average 2.75 tokens per multi-word LE. It should be noted that hyphens were counted as tokens for these counts, indicating that a fair number of the multi-token LEs were hyphenated terms. This revised test and training corpus is distributed.

PERFORMANCE

The test corpus contained 292 sentences, and 5993 LEs. The tagger missed 291 assignments out of the 5993, or 95.1 % correctly assigned tags. While taggers are traditionally compared using this statistic, 95.1% represents only 45.89% of the sentences completely tagged correctly. A further analysis of the failures made show that 64 (21%) of the failures were due to adj/noun and noun/adj assignments – a failure that has less consequence for the task of phrase boundary assignment.

DISCUSSION AND ISSUES

There were surprisingly many instances of overlapping lexical elements – about 100 or so within the test corpus of over 150,000 tokens. Nearly all involved the tokens 'beta', 'cell', 'line', 'protein' and 'binding'. The issue is that both 'beta cell' and 'cell line' are LEs in the lexicon. The greedy lexical lookup will always keep together the first tokens seen, whether this is correct or not. In such an environment, where the distinctions matter, it might be better to use the lexical lookup algorithm that uses shortest spanning match to avoid the conflict. All of the instances of overlapping LE seen were in larger conglomerations of adjective noun sequences within the same phrase. This problem does not arise unless the overlapping LEs have parts of speech that span across phrase boundaries, or the proper identification of phrase constituents is required.

The decision to exclusively hand annotate text from tags contained in the lexicon had to be slightly relaxed when it was observed that there were some odd usages of words within the training and test corpus. The corpus was from the Genomics domain, and contained gene names such as "fixed", and "patched" along with "slouch". These are words that would normally be considered verbs, but were seen to be used as uncount nouns. It is just such cases that justify a local lexicon. These terms could be added (with caution) when their usage differs drastically from their expected usage.

Within the context of the training corpus, it has been argued that multi-word LEs should be annotated at the word level, with tags that indicated that this is part of a larger constituent. The original MedPost corpus contained such tags, indicated as such with a + beside the tag. The hypothesis was that more would be learned from the training, with the added benefit of the constituent parts tagged. Time did not allow for such an experiment.

FUTURE WORK

Training with Untagged Text

The cost, in terms of time and money, to hand annotate a corpus large enough to achieve good performance is prohibitive. It has been reported that, as an alternative, training on a small hand annotated corpus can be supplemented with unsupervised training on a large un-annotated corpus [7]. An unsupervised trainer that updates the HMM by looking at a large amount of untagged text is under development, but was not ready as of yet. It is assumed that the trainWithTags algorithm has previously been run with at least a small amount of tagged text to create an initial HMM.

More work is to be done with shape identification beyond what was already mentioned. Additional shape recognition patterns including a gene name lookup should be made available as an option. Shape patterns also under consideration include recognizing "x-binding" and "x-induced" adj patterns where X may or may not be a recognized word.

The task of reviewing, correcting and adding hand annotations could have been made much easier had a GUI tool been used that would provide potential parts of speech from the Lexicon. Work should be done to seek out and integrate an existing GUI based annotation tool for this purpose.

ACKNOWLEDGEMENTS

The authors wish to thank and recognize the contributions made by Ms. Destinee Tormey and Ms. Ying He.

This work was supported by the Intramural Research Program at NIH, NLM.

REFERENCES

[1] http://SPECIALIST.nlm.nih.gov/theTextTools
[2] http://SPECIALIST.nlm.nih.gov/lexicon
[3] Browne AC, McCray, AT, Srinivasan S. The SPECIALIST Lexicon Technical Report, 6/2000, http://specialist.nlm.nih.gov/lexicon/docs/techrpt.pdf
[4] Smith L, Rindflesch T, Wilbur WJ. MedPost: a part-of-speech tagger for bioMedical text Bioinformatics. 2004 Sep 22;20(14):2320-1.
[5] Manning CD, Schütze H. Foundations of Statistical Natural Language Processing, 2003 Massachusetts Institute of Technology, Chapter 10.
[6] http://SPECIALST.nlm.nih.gov/dtagger/markov.html
[7] Cutting D, Kupiec J, Pedersen J, Sibun P. A Practical Part-of-Speech Tagger, D. Cutting, J. 1992, Proceedings of the Third Conference on Applied Natural Language Processing

Address for correspondence
Guy Divita
National Library of Medicine
8600 Rockville Pike
Bethesda, MD 20894
divita@nlm.nih.gov