# An Alternative Approach to Part-of-Speech Tagging

Daniel Bishop, Indiana University

2012

# The Big Idea

- Statistically-based model
- Eschew traditional left-to-right procedure
- Work off of “seeds” from known words
- HMM-esque bidirectional tagging

# Training the Statistical Model

- Using the Penn Treebank, bigrams and unigrams are extracted.
- Unigrams are separated into unambiguous lexical items and ambiguous lexical items regardless of number of occurrences.
- Bigrams of POS combos and POS-Word combos are created and stored in files as well.

# Unigrams

- Unambiguous tokens get their own file, and they need only exist once in the file. not treated differently due to more instances.
- “dinosaurs” is always NNS, 2.4375 is always CD
- Ambiguous unigrams have the tags and number of occurrences of each tag with that word.
- “Test” (vb 15) (vbp 2) (nn 112) (nnp 4)

# Bigrams

- POS bigrams are separated into individual files by initial letter of the left and right tag and mark the number of occurrences. e.g., (‘jjr’,’dt’) 27
- Combined bigrams include lexical and tag information on a line, organized into files for each lexical initial letter for both left and right portions of the bigram

# Tagging

- Tagging is done in two main stages.
- 1) Tagging of all of the unambiguous tokens in the text. This portion is trivial.
- 2) Tagging in both directions from these “seed” words. This portion is the actual statistically-based tagging and is done in cycles.
- Note the similarity to (Tsuruoka, Tsujii 2005) which tags the “easiest” tag first and can use future as well as past contexts at times.

# Tagging, continued

- Ambiguous tagging actually starts by finding an untagged word and using its surrounding context (if that context is already tagged.)
- The program reads the relevant portion of the POS and combined bigram files, and creates a confidence dictionary of all possible POS tag combos and lexical possibilities.

# Tagging, example

- “hour tour”, with “hour” already tagged as NN
- Tagger looks for instances of “hour tour” and takes note of the possible tags for “tour”. It finds one example of NN. Existing word-word combos get heavy weights to their values.
- Tagger looks for all tags that immediately follow NN, and how many times they show up. NN JJS, NN CD, NN RBR, etc
- Also looks for what tags “hour” gets on its own.

# Tagging example continued

- After gathering all possibilities for left and/or right words and unigram possibilities and POS bigrams, the weights are considered.
- Without the left/right words and POS info, it relies on the unigram ambiguity info and chooses if one tag is used over 95% of the time.
- Otherwise, uses 90% threshold for tagging.

# Tagging, continued

- After going through a file and adding new tags along the way, the tagger will likely have many words left untagged. It then repeats the ambiguous tagging (now able to use context of the newly tagged words) in cycles as specified by the user. Default is until no new tags are assigned, often about 6 cycles.

# Results

- Tagging Section 23 of the Penn Treebank
- Trained on Sections 00-22 and 24 of the PTB
- Cycle 7 (each cycle taking approx. 11.5 min)
- Exactly 45000 words tagged (of 59100) 75.14% tagged
- 43909 correct, 1091 incorrect compared to gold standard
- 97.58% accuracy (compared to TnT's 97.0%)

# Notes

- The tagger is, currently, incomplete. Unknown word handling is yet to be implemented. As the tagger will strive to be language-independent, a fairly robust morphological analyzer must be implemented.
- Primary disadvantage so far is speed – this tagger is orders of magnitude slower than other systems such as TnT or SVMTool. One cycle for Section 23 of the PTB is about 11.5 minutes.

# Future Work

- First will be a system to ensure all known words eventually get tags. If no confidence dictionary is made for a word it is skipped. Other words never get high enough confidence values. Various techniques (look-ahead, threshold altering, etc) will be employed.

# Future Work

- Second will be a morphological analyzer to include during training that, when tagging, can be used to identify unknown words. Suffix analysis similar to (Samuelsson 1993) and a homebrew morphological “chunker” will be used. Current interest especially lies in morphologically rich languages such as Swahili. (Ninasoma → Ni-na-soma → 1SG-PRES-STUDY)

# Future Work

- Lastly will be rounds of speed improvements and usability, compatibility with other types of treebanks, and ability to pass individual sentences as input for incorporation into other programs.
- Work will be made freely available and open-source. Tagger written in Python and highly commented. (You can have it, too! Soon.)

# Future Work

- This tagger will be the basis of an initial stage of a parser of some kind (likely a constituency parser) that I will be making in the months to come.
- Depending on performance of morphological analyzer, may look into machine translation of Swahili <-> English

Questions? Comments?