POS Tagging

Part-Of-Speech Tagging or generally known as POS tagging is the task of tagging words of sentences with a tag to explain their importance or contribution to the speech. This helps in understanding the underlying structure of speech delivered in any language.

The importance of POS tagging a sentence is to efficiently translate from source to target language without eliminating the context or important actions. One such application is to disambiguate homonyms. For instance:

I fish a fish

The same sentence in French would be Je pêche un poisson. Without tagging, fish would be translated the same way in both case, which would lead to a wrong translation. However, after POS tagging, the sentence would be:

I_PRON fish_VERB a_DET fish_NOUN

This would help in translating the sentence much more accurately.

Another important aspect of POS tagging is to derive the action in speech controlled applications. Interpreting the verb in a speech is essential for performing the user requested operation in such applications.

POS Tagging can be done in two ways: Rule based algorithms or stochastically.

Unsupervised POS Tagging: It is a low cost alternative and generally paves the way for supervised methods like Named Entity Recognition and Word Sense Disambiguation.

The use of Unsupervised tagging is important though when very few few training examples exist and adequate performance is expected. Although, tagging the features of a supervised technique in an unsupervised manner is the best way to perform POS tagging as inferred from the comparisons in Paper_3(attached).

After this inference and understanding the analysis of several corpora using both unsupervised and supervised methods, I can conclude that a combinatorial approach works best to achieve a high level of accuracy.

Furthermore, a research into supervised POS tagging methods lead to the introduction of a machine learning concept called **Perceptron**. The approach is simple wherein we include feature weights and learn these weights based on the the correctness of its classification into a set of classes.

After a thorough understanding of the discriminative approach used for POS tagging in Paper_1(attached), I also inferred that a regular perceptron lacks competitiveness in its approach. If we train it on a slightly different data, we have a separate model altogether and therefore, the algorithm lacks generalization.

Therefore, an **Averaged Perceptron** makes the weights more sticky by getting the average of all weights rather than the final weights. This an implementation I'm keen on making and would like to discuss it further with you.

I also went through an approach that allows us to train a simple model using the **Wiktionary** corpora and the model does achieve a great amount of accuracy over other complex learning algorithms for a great number of languages.

This supervised approach as show in Paper_2 (attached) uses other useful information of the Wiktionary corpora such as glosses and translations. The model accuracy comes second to the Stanford tagger when tested on the Brown Corpus and is quite impressive when compared with the HMM and Unsupervised techniques. Counting a major drawback, the error breakdown shows that out of vocabulary is the maximum set encountered over majority of languages but as the wiktionary grows, such errors are likely to come down. From an implementation perspective this would be a big ask within the time limit for this semester, but definitely an interesting topic to explore.

The first few observations after going through supervised and unsupervised approaches of tagging the Brown Corpus were as follows:

For a case-sensitive text, we should incorporate separate features to avoid over fitting to the conventions of the training domain. Although for a robust tagger, we can focus on features such as the frequency of title cased words.

Filtering out the higher frequency words that have unambiguous tags, reduces the computational overhead in the training data set. Roughly, 50% of the corpus can be directly tagged using such filtering.

There have been a few libraries developed with neat documentation to tag a set of text. A Cython greedy tagger achieves 97% accuracy on 130K words of the WSJ.

On the other hand, NLTK and Pattern libraries are robust with rich documentation but have the additional baggage of implementation due to massive frameworks and wrappers involved.

In order to derive a trade off between efficiency and accuracy, tagging can be thought of as a "supervised learning problem". The question to solve is that given a column of words with tags, we need to predict the tag of the word at the ith position.

Averaged Perceptron approach:

The approach involves a set of features along with an associated weight for each feature. In order to process the text, we receive a set of features and a POS tag pair. Using the current weights of these features, we predict the appropriate POS tag.

A few codes snippets are as follows:

Train:

```
def train(self, sentences, save_loc=None, nr_iter=5, quiet=False):
    '''Train a model from sentences, and save it at save_loc. nr_iter
    controls the number of Perceptron training iterations.'''
    self._make_tagdict(sentences, quiet=quiet)
    self.model.classes = self.classes
    prev, prev2 = START
    for iter_ in range(nr_iter):
        c = 0; n = 0
        for words, tags in sentences:
            context = START + [self._normalize(w) for w in words] + END
            for i, word in enumerate(words):
                guess = self.tagdict.get(word)
                if not guess:
                    feats = self._get_features(
                            i, word, context, prev, prev2)
                    guess = self.model.predict(feats)
                    self.model.update(tags[i], guess, feats)
                # Set the history features from the guesses, not the
                # true tags
                prev2 = prev; prev = guess
                c += guess == tags[i]; n += 1
```

```
        random.shuffle(sentences)
        if not quiet:
            print("Iter %d: %d/%d=%.3f" % (iter_, c, n, _pc(c, n)))
    self.model.average_weights()
    # Pickle as a binary file
    if save_loc is not None:
        cPickle.dump((self.model.weights, self.tagdict, self.classes),
                     open(save_loc, 'wb'), -1)
```

Features:

```
def _get_features(self, i, word, context, prev, prev2):
    '''Map tokens-in-contexts into a feature representation, implemented as a
    set. If the features change, a new model must be trained.'''
    def add(name, *args):
        features.add('+'.join((name,) + tuple(args)))

    features = set()
    add('bias') # This acts sort of like a prior
    add('i suffix', word[-3:])
    add('i pref1', word[0])
    add('i-1 tag', prev)
    add('i-2 tag', prev2)
    add('i tag+i-2 tag', prev, prev2)
    add('i word', context[i])
    add('i-1 tag+i word', prev, context[i])
    add('i-1 word', context[i-1])
    add('i-1 suffix', context[i-1][-3:])
```

```
    add('i-2 word', context[i-2])
    add('i+1 word', context[i+1])
    add('i+1 suffix', context[i+1][-3:])
    add('i+2 word', context[i+2])
    return features
```

Weight Update:

```
def update(self, truth, guess, features):
    def upd_feat(c, f, v):
        nr_iters_at_this_weight = self.i - self._timestamps[f][c]
        self._totals[f][c] += nr_iters_at_this_weight * self.weights[f][c]
        self.weights[f][c] += v
        self._timestamps[f][c] = self.i

    self.i += 1
    for f in features:
        upd_feat(truth, f, 1.0)
        upd_feat(guess, f, -1.0)
```

Universal Part of Speech Tagging

The second approach is to exploit the common linguistic phenomena of languages and obtain a universal set for part of speech tags. There has been an exhaustive linguistic research already done for obtaining mappings of such languages. The idea of using a common set of tags decreases the complexity considerably.

The paper by Petrov, Das & McDonald on this topic is of considerable value as it exploits such common features across 25 languages and obtains a great deal of accuracy in POS Tagging across a common corpus.

The results are as follows:

Language	Source	# Tags	O/O	U/U	O/U
Arabic	PADT/CoNLL07 (Hajič et al., 2004)	21	96.1	96.9	97.0
Basque	Basque3LB/CoNLL07 (Aduriz et al., 2003)	64	89.3	93.7	93.7
Bulgarian	BTB/CoNLL06 (Simov et al., 2002)	54	95.7	97.5	97.8
Catalan	CESS-ECE/CoNLL07 (Martí et al., 2007)	54	98.5	98.2	98.8
Chinese	Penn ChineseTreebank 6.0 (Palmer et al., 2007)	34	91.7	93.4	94.1
Chinese	Sinica/CoNLL07 (Chen et al., 2003)	294	87.5	91.8	92.6
Czech	PDT/CoNLL07 (Böhmová et al., 2003)	63	99.1	99.1	99.1
Danish	DDT/CoNLL06 (Kromann et al., 2003)	25	96.2	96.4	96.9
Dutch	Alpino/CoNLL06 (Van der Beek et al., 2002)	12	93.0	95.0	95.0
English	PennTreebank (Marcus et al., 1993)	45	96.7	96.8	97.7
French	FrenchTreebank (Abeillé et al., 2003)	30	96.6	96.7	97.3
German	Tiger/CoNLL06 (Brants et al., 2002)	54	97.9	98.1	98.8
German	Negra (Skut et al., 1997)	54	96.9	97.9	98.6
Greek	GDT/CoNLL07 (Prokopidis et al., 2005)	38	97.2	97.5	97.8
Hungarian	Szeged/CoNLL07 (Csendes et al., 2005)	43	94.5	95.6	95.8
Italian	ISST/CoNLL07 (Montemagni et al., 2003)	28	94.9	95.8	95.8
Japanese	Verbmobil/CoNLL06 (Kawata and Bartels, 2000)	80	98.3	98.0	99.1
Japanese	Kyoto4.0 (Kurohashi and Nagao, 1997)	42	97.4	98.7	99.3
Korean	Sejong (http://www.sejong.or.kr)	187	96.5	97.5	98.4
Portuguese	Floresta Sintá(c)tica/CoNLL06 (Afonso et al., 2002)	22	96.9	96.8	97.4
Russian	SynTagRus-RNC (Boguslavsky et al., 2002)	11	96.8	96.8	96.8
Slovene	SDT/CoNLL06 (Džeroski et al., 2006)	29	94.7	94.6	95.3
Spanish	Ancora-Cast3LB/CoNLL06 (Civit and Martí, 2004)	47	96.3	96.3	96.9
Swedish	Talbanken05/CoNLL06 (Nivre et al., 2006)	41	93.6	94.7	95.1
Turkish	METU-Sabanci/CoNLL07 (Oflazer et al., 2003)	31	87.5	89.1	90.2

As a part of implementing such a mapping, I had built a speech to POS Tagger using Python, NLTK and Speech API's. The files have been provided as SpeechPosTagging.zip. The idea was to use these mappings and generate smoother translations by obtaining a universally tagged set of text.

Incremental Improvements:

A few errors that still crop up in the Penn Treebank, used by the Stanford NLP Tagger operated on WSJ are:

the/DT largest/JJS newsweekly/RB , had average circulation of
Correct: newsweekly/NN

below the $ 2.29 billion value United Illuminating places/NNS on its bid
Correct: places/VBZ

Rowe also noted that political concerns also worried/VBN New England Electric .
Correct: worried/VBD

Commonwealth Edison now faces an additional court-ordered refund on its summer/winter rate differential collections that/IN the Illinois Appellate Court has estimated at $ 140 million . Correct: that/WDT

Joseph/NNP M./NNP Blanchard/NNP , 37 , vice president , engineering ; Malcolm/NNP A./NN Hammerton/NNP
Correct: A./NNP

The remaining errors are going to be solved by better local features of the kind used by current state-of-the-art sequence model taggers. An analysis of such errors is given as follows:

Class	Frequency
1. Lexicon gap	4.5%
2. Unknown word	4.5%
3. Could plausibly get right	16.0%
4. Difficult linguistics	19.5%
5. Underspecified/unclear	12.0%
6. Inconsistent/no standard	28.0%
7. Gold standard wrong	15.5%

In order to overcome such errors, the following features need to be looked upon as incremental improvements:

Past tense versus past participles:

In general, if a past participle is not adjacent to a passive or perfective auxiliary indicating a VBN, then it is quite frequently wrongly tagged as VBD.13 But such cases can usually be detected and fixed by rules over tree patterns. For instance, we can use rules such as this one: @VP < VBD=bad [> (@VP < (/^VB/ < be have get)) | > (@VP < CONJP|CC >

(@VP < (/^VB/ < be have get))) | > (@NP < @NP)] relabel bad VBN. That is, a verb in a VP that is under a VP containing a passive or perfective auxiliary verb (perhaps inside a conjunction structure) or modifying a noun phrase should really be a participle VBN and not a past finite VBD.

Plurals as singulars

Words like United Airlines and United States get treated as plurals but these are exceptional cases. In order to overcome such plurals the following rule is suggested:

NNP=bad < Industries|Airlines relabel bad NNPS

Tagging "That"

1. @NP < (IN|WDT=bad < /^(?:a|that|That)$/)
   relabel bad DT
2. @SBAR < (DT|WDT|NN|NNP|RB=bad < that|because|while|Though)
   relabel bad IN
3. @ADJP < JJ < (IN=bad < that)
   relabel bad RB

The first rule matches 173 times in the Penn Treebank, while the second rule matches 285 times.

References:

Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments
Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills,
Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith
School of Computer Science, Carnegie Mellon Univeristy, Pittsburgh, PA 15213, USA

A Universal Part-of-Speech Tagset
Slav Petrov, Google Research, New York, NY, USA, slav@google.com
Dipanjan Das, Carnegie Mellon University, Pittsburgh, PA, USA, dipanjan@cs.cmu.edu
Ryan McDonald, Google Research, New York, NY, USA, ryanmcd@google.com

Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?
Christopher D. Manning, Departments of Linguistics and Computer Science, Stanford
University, 353 Serra Mall, Stanford CA 94305-9010, manning@stanford.edu

A Maximum Entropy Model for Part of Speech Tagging
Adwait Ratnaparkhi, University of Pennsylvania, Dept. of Computer Science,
adwait@gradient.cis.upenn.edu

Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments
Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills,
Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith
School of Computer Science, Carnegie Mellon Univeristy, Pittsburgh, PA 15213, USA

Tagging Accuracy Analysis on Part-of-Speech Taggers
Semih Yumusak[1], Erdogan Dogdu[2], Halife Kodaz[3]
Computer Engineering Department, KTO Karatay University, Konya, Turkey

Learning a Part-of-Speech Tagger from Two Hours of Annotation
Dan Garrette, Dept. of Computer Science, The University of Texas at Austin, dhg@cs.utexas.edu
Jason Baldridge, Dept. of Linguistics, The University of Texas at Austin, jbaldrid@utexas.edu

Discriminative Training with Perceptron Algorithm for POS Tagging Task
Mahsa Yarmohammadi, Center for Spoken Language Understanding, Oregon Health & Science
University, Portland, Oregon, yarmoham@ohsu.edu

Wiki-ly Supervised Part-of-Speech Tagging
Shen Li, Computer & Information Science, University of Pennsylvania, shenli@seas.upenn.edu
Ben Taskar, Computer Science, University of Pennsylvania, taskar@cis.upenn.edu

Unsupervised Part-Of-Speech Tagging Supporting Supervised Methods
Chris Biemann, University of Leipzig, NLP Dept., Johannisgasse 26, 04103, Leipzig, Germany,
biem@informatik.uni-leipzig.de