POS tagging of twitter

Venugopal Gonela
11081500

Problem statement:

With the growing popularity of microblogging service like twitter, a good source of user content is created. Much work is being done to analyze and exploit that data. The fundamental part of NLP pipeline is POS tagging, which is a basic form of syntactic analysis. I would like to tag tweets with HMM POS tagger and find relevant features of tweets to improve its performance. The input to the tagger would be tweets and the output would be all the tokens of the tweet with appropriate POS tag for each token.

Dataset:

The data used in this project is twitter data that consists of 1,827 English tweets (26,436 tokens). The data is divided into 1,000 tweets as train set for the HMM tagger, 327 tweets for development set and 500 tweets for test set. The dataset is obtained from https://github.com/brendano/ark-tweet-nlp. The dataset is tagged with 25 coarse grained tags. The tagset is explained in the following figure [1].

Tag	Description	Examples	%
Nominal, Nominal + Verbal			
N	common noun (NN, NNS)	books someone	13.7
O	pronoun (personal/WH; not possessive; PRP, WP)	it you u meeee	6.8
S	nominal + possessive	books' someone's	0.1
^	proper noun (NNP, NNPS)	lebron usa iPad	6.4
Z	proper noun + possessive	America's	0.2
L	nominal + verbal	he's book'll iono (= *I don't know*)	1.6
M	proper noun + verbal	Mark'll	0.0
Other open-class words			
V	verb incl. copula, auxiliaries (V*, MD)	might gonna ought couldn't is eats	15.1
A	adjective (J*)	good fav lil	5.1
R	adverb (R*, WRB)	2 (i.e., *too*)	4.6
!	interjection (UH)	lol haha FTW yea right	2.6
Other closed-class words			
D	determiner (WDT, DT, WP$, PRP$)	the teh its it's	6.5
P	pre- or postposition, or subordinating conjunction (IN, TO)	while to for 2 (i.e., *to*) 4 (i.e., *for*)	8.7
&	coordinating conjunction (CC)	and n & + BUT	1.7
T	verb particle (RP)	out off Up UP	0.6
X	existential *there*, predeterminers (EX, PDT)	both	0.1
Y	X + verbal	there's all's	0.0
Twitter/online-specific			
#	hashtag (indicates topic/category for tweet)	#acl	1.0
@	at-mention (indicates another user as a recipient of a tweet)	@BarackObama	4.9
~	discourse marker, indications of continuation of a message across multiple tweets	RT and : in retweet construction RT @user : hello	3.4
U	URL or email address	http://bit.ly/xyz	1.6
E	emoticon	:-) :b (: <3 o_O	1.0
Miscellaneous			
$	numeral (CD)	2010 four 9:30	1.5
,	punctuation (#, $, '', (,), ,, ., :, ``)	!!! .... ?!?	11.6
G	other abbreviations, foreign words, possessive endings, symbols, garbage (FW, POS, SYM, LS)	ily (*I love you*) wby (*what about you*) 's ♫ --> awesome...I'm	1.1

Figure1: Tagset used for twitter data [1].

Implementation details:

I have implemented bi-gram Hidden Markov Model(HMM) for part of speech tagging(POS) of the tweets. It uses training data to estimate emission or word-tag probabilities and transition probabilities.

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i) \qquad P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

Emission Probability Transition Probability

Given a sequence of words to be tagged, using Bayes theorem, the best possible sequence of tags for given words is given by

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i|t_i)}^{\text{emission}} \overbrace{P(t_i|t_{i-1})}^{\text{transition}}$$

The sequence of tags for words are calculated efficiently using Viterbi algorithm. I have implemented the Viterbi algorithm and estimated the probabilities using maximum likelihood method from training data using python programming language. A lot of preprocessing is done on the words to get a very good accuracy of POS tagging. All the preprocessing is also implemented using the python language.

Experiment details:

Baseline: Initially I have developed a baseline tagger, which tags each word in the tweets of test set to the most probable tag from the training data. After doing the preprocessing on the words, the accuracy is shown as **75.02%.** The preprocessing details of the words is discussed in the following section of HMM.

HMM: Using bi-gram HMM tagger, the accuracy is drastically increased to **86.41**% by writing the morphological rules for new words which show up in test data. I will discuss the preprocessing used on the words.

Preprocessing:

- There is a special tag defined for URLs and email addresses. Hence, converted all URLs and email-addresses to <url> word.
- There is a special tag defined for Twitter hashtags. Hence, I have converted all hashtags to <#>.
- There is a special tag defined for emoticons. Hence, I have converted all possible emoticons in the data to <emot>.
- There is a special tag defined for all Twitter at-mentions, of the form @user, which link to other Twitter users from within a tweet. Hence, I converted all at-mentions to <@>
- All the digits and digit like words are converted to <num>. For example, words like 12:20, one, million, 75mph, 20s etc. are all converted to <num>.
- Repetitive characters in words are stemmed to one character. For example, word like reallllllyyyyyyy is changed to really. The words show the intense emotion of the user but have same POS tag irrespective of the emotion used in the word.
- All words are lower cased before estimating emission and transition probabilities.

The set of 1,827 annotated tweets are divided into a training set of 1,000 (14,620 tokens), a development set of 327 (4,823 tokens), and a test set of 500 (7,152 tokens). While testing, the probability of unknown word w, $P(w|t_i)$ will be zero. But $P(t_i|w)$ can be estimated using morphological rules of the unknown word w or by using a method which will be discussed in detail in next section. Then $P(w|t_i)$ can be calculated in the following way.

$$P(w|t_i) = \frac{P(t_i|w) \times P(w)}{P(t_i)}$$

The total number of unknown tokens seen in validation set which are not present in training set is 1,459. The probability of unknown tokens tagger can see is generalized as 0.091. By equally distributing this probability among all unknown tokens, the probability of each unknown word can be easily obtained.

The following are results before and after doing the preprocessing on the both baseline and HMM taggers.

Before preprocessing:

Tagger/Data	Training set	Dev set	Testing set
Baseline	94.97	62.87	64.28
HMM	**97.21**	**69.87**	**69.28**

After preprocessing:

Tagger/Data	Training set	Dev set	Testing set
Baseline	92.95	75.02	75.13
HMM	**95.56**	**80.95**	**81.43**

Note: In HMM, the tag given to the unknown word is Noun(N).

It is clearly shown above that, preprocessing the tokens has drastically increased the accuracy of both baseline taggers and HMM tagger. The HMM has outperformed the baseline with a significant amount. It is because we are generalizing all numbers, URLs, at-mentions, hashtags, emoticons to each unique word which reduces the unknown words in the dev and test sets drastically.

Morphological rules for unknown words:

By analyzing the tokens in the validation set, I have added few morphological rules to assign more probability to more likely tags for that unknown word. I have also used the corpus given in course assignment 3 of POS tagging and converted that corpus to the tagset of twitter and used it to estimate probability of tag given unknown word. I will show how the tagger increased its performance by adding each morphological rule and also using the probabilities estimated from general English corpus.

By analyzing the errors made on the development set, the percentage is calculated for tags that are wrongly predicted as shown below.

1) P(t='N'|word) = 0.5, P(t='^'|word) = 0.4 and P(t='V'|word) = 0.1, the accuracy is increased as follows,

Tagger/Data	Training set	Dev set
HMM	95.56	**83.75 ↑2.8**

2) If word ends with '**ed**', then P(t='V'|word) = 0.5, P(t='A'|word) = 0.4 and P(t='N'|word) = 0.1, the accuracy is increased as follows

Tagger/Data	Training set	Dev set
HMM	95.56	**83.97 ↑0.22**

3) If word ends with '**ing**', then P(t='V'|word) = 0.7, P(t='N'|word) = 0.2 and P(t='A'|word) = 0.1, the accuracy is increased as follows

Tagger/Data	Training set	Dev set
HMM	95.56	**84.31 ↑0.34**

4) If word ends with '**ly**', then P(t='R'|word) = 0.7, P(t='A'|word) = 0.2, P(t='V'|word) = 0.05 and P(t='G'|word) = 0.05 the accuracy is increased as follows

Tagger/Data	Training set	Dev set
HMM	95.56	**84.43 ↑0.12**

5) If word ends with **''s**', then P(t='S'|word) = 0.3, P(t='Z'|word) = 0.3, P(t='L'|word) = 0.2, P(t='N'|word) = 0.05 and P(t='^'|word) = 0.05 the accuracy is increased as follows

Tagger/Data	Training set	Dev set
HMM	95.56	**84.54 ↑0.11**

Please note that the likelihood of a tag to the word shape is calculated based on the mistakes the tagger made on development set and mistakes are then grouped based on the word shape.

In process of improving the accuracy even more, I have estimated probabilities of tag given word from different corpus of English language which was given in course assignment 3. Then used those estimates to calculate probability of unknown word given tag on twitter data. The following is the improvement in accuracy.

Tagger/Data	Training set	Dev set
HMM	95.56	**86.71 ↑2.17**

The HMM bigram tagger which is developed as explained above is tested on the testing data. The following are the results shown, and also compared with the baseline tagger.

Tagger/Data	Training set	Dev set	Testing set
Baseline	92.95	75.02	75.13
HMM	**95.56**	**86.71**	**86.41**

The English corpus used to estimate the unknown word probability has really helped in improving the accuracy of the system. Also the results show that HMM tagger has improved drastically from the baseline tagger.

Error Analysis:

Error analysis is made on the test set, by calculating accuracy rates per tag and most confused tag for each tag. The following are the results.

Tag	Count	Accuracy	Confused with		Tag	Count	Accuracy	Confused with
V	1053	89.9	N		!	186	71.5	^
N	981	81.5	^		L	129	85.3	^
,	880	96.4	~		&	127	97.6	^
P	616	92.4	R		U	117	97.4	N
O	495	94.1	D		$	85	88.2	^
D	449	93.3	!		#	78	92.3	^
^	495	68.1	N		G	70	20	^
A	367	72.2	N		E	63	82.5	,
R	339	81.4	P		T	36	69.4	P
@	330	99.1	!		Z	22	45.4	^
~	212	86.3	,		S	6	50	Z

The table gives the information of tags it is most confused with. Most unknown words are of open class categories and are often confused with other open class categories. The common noun (N) and proper noun (^) are very difficult to tag as they are very difficult to differentiate.

Conclusion:

I have developed a part-of-speech bigram tagger using HMM Viterbi algorithm. The accuracy is improved using other English corpus other that twitter corpus alone. All the details are discussed how each morphological rule has improved its performance. Error analysis has made to show what tags are difficult to tag and with what tags they are confused with.

References:

1. Kevin Gimpel, Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments.