Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Joint POS Tagging and Text Normalization for Informal Text

Chen Li and Yang Liu
University of Texas at Dallas
Richardson, TX, 75080, USA
{chenli,yangl}@hlt.utdallas.edu

Abstract

Text normalization and part-of-speech (POS) tagging for social media data have been investigated recently, however, prior work has treated them separately. In this paper, we propose a joint Viterbi decoding process to determine each token's POS tag and non-standard token's correct form at the same time. In order to evaluate our approach, we create two new data sets with POS tag labels and non-standard tokens' correct forms. This is the first data set with such annotation. The experiment results demonstrate the effect of non-standard words on POS tagging, and also show that our proposed methods perform better than the state-of-the-art systems in both POS tagging and normalization.

1 Introduction

There has been a rapid increase in social media text in the last few years, including the mobile phone text message (SMS), comments from the social media websites such as Facebook and Twitter, and real-time communication platforms like MSN and Gtalk. Unfortunately, traditional natural language processing (NLP) tools sometimes perform poorly when processing this kind of text. One reason is that social text is very informal, and contains many misspelled words, abbreviations and many other non-standard tokens.

There are several ways to improve language processing performance on the social media data. One is to leverage normalization techniques which can automatically convert the non-standard tokens into the corresponding standard words. Intuitively this will ease subsequent language processing modules for the domain of social media that contains non-standard tokens. For example, if '2mr' is converted to 'tomorrow', a text-to-speech system will know how to pronounce it, a POS tagger can label it correctly, and an information extraction system can identify it as a time expression. This task has received increasing attention in social media language processing. Another way is to design special models and data or apply specific linguistic knowledge in this domain [Ritter *et al.*, 2011; Owoputi *et al.*, 2013; Ritter *et al.*, 2010; Foster *et al.*, 2011; Liu *et al.*, 2011; 2012]. For example, the system in [Ritter *et al.*, 2011] reduced the POS tagging prediction error by 41% compared with the Stanford POS Tagger, and by 22% in parsing tweets compared with the OpenNLP chunker tool. [Owoputi *et al.*, 2013] created a new set of POS tags for Twitter data, and showed improved POS tagging performance for such data when using their tag set and word cluster information extracted from a huge Twitter corpus.

In this paper, our objective is to perform POS tagging and text normalization at the same time. Through analysis of previous POS tagging results in social media data, we find that non-standard tokens indeed have a negative impact on POS tagging. They affect not only the accuracy of the POS tags for themselves, but also their surrounding correct words because context information is an very important feature for POS tagging. Therefore, we expect that explicitly performing normalization would improve POS tagging accuracy, for both the non-standard words themselves and their context words. On the other hand, previous work in normalization mostly used word level information, such as character sequences and pronunciation features. Some work [Yang and Eisenstein, 2013; Li and Liu, 2014] leveraged unsupervised methods to construct the semantic relationship between non-standard tokens and correct words by considering words' context. Deeper linguistic information such as POS tags has not been incorporated for normalization. Motivated by these, we propose to jointly perform POS tagging and text normalization, in order to let them benefit each other. Although joint learning and decoding approaches have been widely used in many tasks, such as joint Chinese word segmentation and POS tagging task [Zhang and Clark, 2008], joint text sentence compression and summarization [Martins and Smith, 2009], this is the first time to apply joint decoding for normalization and POS tagging in informal text.

Therefore, our work is related to POS tagging, a fundamental research problem in NLP, which has countless applications. One of the latest research on POS tagging in social media domain is from [Gimpel *et al.*, 2011; Owoputi *et al.*, 2013]. They built a POS tagger for tweets using 25 coarse-grained tags and also provided two date sets annotated with this special tag set. Then they incorporated unsupervised word cluster information into their tagging system and achieved significant improvement on tagging performance. Another line of work closely related to ours is text normalization in social media domain. Lots of approaches have been developed for this task, from using edit distance

[Damerau, 1964; Levenshtein, 1966], to the noisy channel model [Cook and Stevenson, 2009; Pennell and Liu, 2010; Li and Liu, 2012a] and machine translation method [Aw *et al.*, 2006; Pennell and Liu, 2011; Li and Liu, 2012b]. Normalization performance on some bench mark data has been improved a lot.

Our contributions in this paper are as follows: (1) To the best of our knowledge, this is the first time that an effective and joint approach is proposed to combine the normalization and POS tagging techniques to improve the performance of these two tasks on English social media data; (2) We created two data sets for this joint task. In these two data sets, every token is labeled with POS tag and a correct word if it is a non-standard token; (3) We demonstrate the effectiveness of our proposed method. Our results outperform the state-of-the-art POS tagger and normalization systems in two different data sets. Our analysis shows the impact of non-standard words on POS tagging and the effect of various factors in the joint model.

2 Data Set

Since there is no prior work performing joint POS tagging and normalization of non-standard tokens at the same time, we created a data set[1] for such a joint task by reusing previous widely used data for each of these two tasks.

[Owoputi *et al.*, 2013] released two data sets, called OCT27 and DAILY547 respectively. These two data sets are annotated with their designed POS tag set (see their paper for details). All together there are 2374 tweets. We asked six native English speakers (they are also social network heavy users) to find the non-standard tokens from these tweets, and also provide the corresponding correct words according to their knowledge. The annotation results showed that 798 tweets contain at least one non-standard token (The rest 1576 sentences have no non-standard tokens). We put these 798 tweets in one data set, which has both POS tag labels and non-standard tokens with their correct word forms. This data set is called Test Data Set 1 in the following of this paper.

[Han and Baldwin, 2011] released a data set including 549 tweets. Each tweet contains at least one non-standard token and the corresponding correct word. In order to label POS tags for these tweets, we first trained a POS tagging model based on the 2374 tweets mentioned above, using the same features from [Owoputi *et al.*, 2013]. Then we applied this model to these 549 tweets and asked one English native speaker to correct the wrong tags manually. After this procedure, these 549 tweets also have POS tags and non-standard tokens' annotation. We call it Test Data Set 2 in the rest of the paper. Please note that every sentence in these two test sets has at least one non-standard token.

3 POS Tagging and Normalization Baseline

3.1 POS Tagging

[Owoputi *et al.*, 2013] used a Maximum Entropy Markov model (MEMM) to implement a POS tagger for Twitter domain.In addition to the contextual word features, they included other features such as the cluster-based features, a word's most frequent POS tags in Penn TreeBank[2] tags, a token-level name list feature, which fires on words from names from several sources. Please refer to [Owoputi *et al.*, 2013] for more details about their POS tagging system and the features. We built a POS tagger baseline using CRF models, rather than MEMM, but kept the same features as used in [Owoputi *et al.*, 2013].

To help understand our proposed joint POS tagging and normalization in the next section, here we briefly explain the Viterbi decoding process in POS tagging. Figure 1 shows the trellis for part of the test word sequence 'so u should answr ur phone'. Every box with dashed lines represents a hidden state (possible POS tag) for the corresponding token. Two sources of information are used in decoding. One is the tag transition probability, $p(y_i|y_j)$, from the trained model, where y_i and y_j are two POS tags. The other is $p(y_i|t_2, t_1t_2t_3)$, where y_i is the POS label for token t_2, t_1 is the previous word and t_3 is the following word. The reason we drop the entire sequence from the condition is because all the features are defined based on a three-word window.

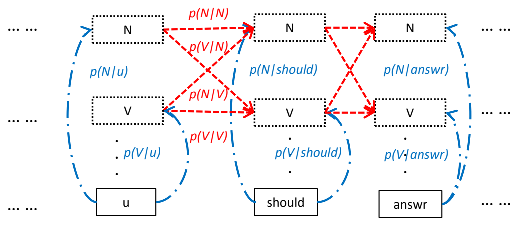

Figure 1: A simple trellis of Viterbi decoding for POS tagging.

3.2 Normalization Model

[Li and Liu, 2014] achieved state-of-the-art normalization performance on Test Data Set 2 by using a reranking technique that combines multiple normalization systems. We use the same normalization method in this study. Briefly, there are three supervised and two unsupervised normalization systems for each non-standard token, resulting in six candidate lists (one system provides two lists). Then a Maximum Entropy reranking model is adopted to combine and rerank these candidate lists, using a rich set of features. After reranking, for each non-standard token t, the system provides a candidate list, and a probability $p(s_i|t)$ for each candidate s_i.

Viterbi decoding is used for sentence level normalization after generating the token level normalization candidates and scores. Figure 2 shows a decoding trellis for normalization. Here for the non-standard tokens, the hidden states represent normalization word candidates. For a standard word that does not need normalization, we can treat it as a special case and use a state with the word itself (the normalization probability is 1 in this case). The scores used in decoding are: the probability of a normalization candidate word s_j given the current

[1]http://www.hlt.utdallas.edu/∼chenli/normalization_pos/

[2]http://www.cis.upenn.edu/ treebank/

token t_i, $p(s_j|t_i)$, which is from the token level normalization model; and the transition probability from a language model, $p(s_j|s_l)$, where s_l is a candidate word for the previous token t_{i-1} (for the standard word case, it will be just the word itself). Note that in the trellis used for normalization, the number of hidden states varies for different tokens. The trellis shown here is for a first-order Markov model, i.e., a bigram language model is used. This is also what is used in [Li and Liu, 2014]. The states can be expanded to consider a trigram language model, which is actually what we will use later in the joint decoding section.

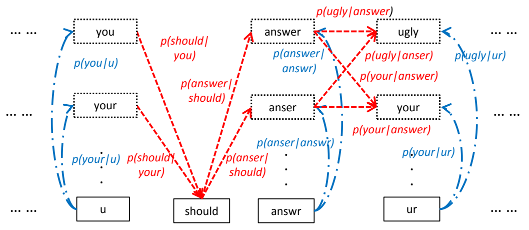

Figure 2: A simple trellis of Viterbi decoding for Normalization

4 Proposed Method

For a given sequence of words, our task is to find the normalized sequence and the POS tags for the words. Rather than using a pipeline method that performs normalization first, and then POS tagging on the normalized sequence, we propose to use joint decoding to predict the normalized words and the POS tags together. This is expected to avoid generating the single best, but likely incorrect normalization hypothesis to be used for POS tagging, as well as be able to leverage POS information to help normalization.

4.1 Joint Decoding for POS and Normalization

Our proposed joint decoding approach for POS tagging and normalization is to combine the above two decoding procedures together as one process. In this joint decoding process, for a token in the test sequence, its hidden states consist of a normalization word candidate and its POS tag. Since for POS tagging and normalization, we use information from the previous and the following word (extracting features in POS tagging, n-gram LM probability in normalization), we put these contextual words explicitly in the states.

Figure 3 shows part of the trellis for the example used previously (test sequence '*u should answr ur*'). It can be thought of as a combination of Figure 1 and Figure 2. Let us assume that each non-standard token ('*u*', '*answr*' and '*ur*') has two normalization candidate words. They are *you* and *your* for *u*, *answer* and *anser* for *answr*, and *ugly* and *your* for *ur*. A black box with dashed lines in Figure 3 represents a state.

There are some properties of this trellis worth pointing out. In joint decoding, each state is composed of a POS tag and normalization word. Furthermore, as mentioned earlier, we include the previous word and the following word in the state (these are the normalization word candidates for the previous and next word). For one state (N, should, you, answer), $p(N|should, you, answer)$ means the probability of word *should*'s POS is Noun, given its previous token's (*u*) normalization is 'you' and next token's (*answr*) normalization is 'answer'. The green box in Figure 3 indicates the same three-word sequence, with different POS tags (one green box corresponds to the states for a token in Figure 1, where the word and context are fixed, and the states are just about POS tags). In this trellis, not all the transitions among the states for two consecutive tokens are valid. Figure 3 shows an example – a path from state (N, should, you, answer) to state (N, anser, should, your) is illegal because they do not share the same word sequence. This is a standard note when using trigram LM in decoding.

Regarding the number of states for each token, it depends on the number of POS tags (k), and the number of normalization word candidates of the current token (l), the previous token (m), and the following token (n). For a standard word, it has just one normalization candidate, the word itself. The number of the hidden states of a token is $l * m * n * k$.

Using such defined states, we can perform Viterbi decoding to find the best state sequence, i.e., the best POS tags and normalization results. The following scores need to be considered to compute the forward score for each state:

- p_1: the probability of the state's POS tag given the three words in this state;
- p_2: the probability of the normalization candidate for the token;
- p_3: the transition probability of the POS tag from last state to that of the current state;
- p_4: trigram language model probability from the previous state to the current state (current word given the previous state's first two words);

The first two probabilities are related to emission probabilities between the hidden state and the observed token, coming from the POS tagger and the normalization model respectively. We use a parameter α when adding them together. Again, for a standard word, its normalization candidate is just itself, and p_2 is 1 in this case. The last two probabilities are about the transition probability between the two states. We use β when combining the POS tag transition probability with the trigram LM probability.

For a hidden state s of an observed token t_i, if its normalization candidate is c_t, its previous and following words are $c_{t_{i-1}}$ and $c_{t_{i+1}}$, and its POS tag is y_j, we define the forward value of this hidden state as following:

$$\begin{aligned} f(s) \quad = \quad & \max_{s' \in \widehat{S}} [\, f(s') + \\ & \alpha * p_1(y_j | c_t, c_{t_{i-1}}\, c_{t_{i+1}}) + p_2(c_t|t_i) + \\ & p_3(y_j|pos(s')) + \beta * p_4(c_t|bigram(s'))] \end{aligned} \tag{1}$$

in which $\widehat{S}$ is the set of all the hidden states of the previous observed token with valid transitions to this current state; function pos returns the state's POS tag, and $bigram$ returns the state's current word and its previous context word. We

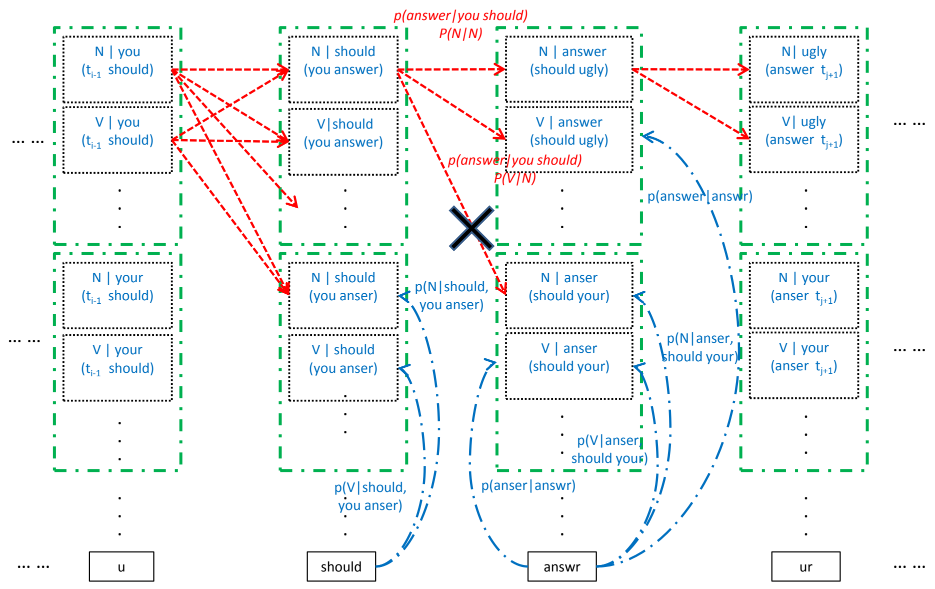

Figure 3: An example trellis for joint decoding.

can use backtracking pointers to recover the highest scoring state sequence.

4.2 Running Time

Assuming the number of normalization candidates for each non-standard token is N (for some words, there may be fewer candidates if the normalization systems cannot provide N candidates), our proposed algorithm's running time is roughly O(LN^4K^2), where L is the length of the sequence, and K is the number of POS tags. This is because there are N^3K hidden states for a non-standard token, and there are NK states from the previous token that have valid transitions to each state of the current token. Of course, this is a worst case scenario. In practice, a sentence has many standard words that do not need normalization, and can significantly reduce the number of states to be considered (N^3 factor becomes 1). Furthermore, pruning can be applied to remove the candidate states for each position. When using the pipeline approach, the normalization decoding's complexity is O(LN^4) (similar reason as for the joint decoding, except that the POS tag is not part of the states when trigram LM is used), and the complexity for POS tagging is O(LK^2).

4.3 Training

In the description above, we assumed that the POS tagger and the normalization model are trained separately, using their own labeled data sets, and the two models are combined in the joint decoding process. However, it is possible to train the two models in a joint fashion, if there is a fully annotated corpus that is labeled with POS tags and non-standard token's correct form. Structured perceptron [Collins, 2002] can be applied to update the weights in each model based on the current results using joint decoding. In addition, this kind of training strategy is also applicable for partially annotated corpus. For example, if a corpus has only normalization label, we can use it to train a normalization model using the joint normalization and POS decoding results.

5 Experiments and Results

5.1 Experiment Setup

To evaluate the impact of normalization on POS tagging, and the benefit of joint decoding on both normalization and POS tagging, we use the following experimental setups.

(a). POS tagging

As we mentioned in Section 2, 798 tweets out of 2,374 are selected as Data Set 1. Therefore when testing on the Data Set 1, we used the rest 1576 tweets with the POS labels as the training data for the CRF POS tagging model, implemented using the Pocket CRF toolkit.When testing on Data Set 2, we use all the 2374 tweets to train the POS tagger.

(b). Normalization

For the normalization model, all the supervised normalization systems are trained using the data released by [Li and Liu, 2014].[3] It has 2,333 unique pairs of non-standard tokens and standard words, which are collected from 2,577 Twitter messages (selected from the Edinburgh Twitter corpus [Petrovic *et al.*, 2010]). This training data has only normalization annotation, not POS information. We first used the Maximum Entropy reranking for token level normalization, and then a sentence level decoding process (introduced in Section 3.2) was

[3]http://www.hlt.utdallas.edu/∼chenli/normalization/

used to generate normalization results.[4] We tried bigram and trigram language models during sentence level decoding.

(c). Normalization + POS tagging

In this experiment, all the tweets are first normalized using the above normalization system in (b) (with only best hypothesis), followed by POS tagging, described in (a).

(d). Oracle Normalization + POS tagging

Here for each non-standard word, we use the reference normalized words. POS tagging is then applied to these normalized tweets. This can be considered as an oracle performance.

(e). Joint Decoding using Separately Trained Models

The two setups above use a pipeline process: normalization followed by POS tagging. Our first joint decoding experiment uses the POS tagger and the normalization models trained independently. Joint Viterbi decoding is used to combine the information from these models to make the final prediction. The number of non-standard token's candidates is set as 10, and parameter α and β are both set as 0.8.

(f). Joint Decoding using Partial Jointly Trained Model

This one also uses joint decoding; however, we apply perceptron training strategy and joint decoding process to train a normalization model, while keeping the POS model fixed. We could also use this strategy to train a POS tagging model if the 1576 tweets with POS labels also have non-standard tokens. In addition, we could simultaneously train both models if we have the corpus with both labels. However, data with such annotations is quite limited (they are used as our test sets). Therefore, we leave the fully joint training task for future research.

5.2 Experiment Results

Table 1 shows the POS tagging and normalization accuracies using different setups. Note that we assume we know which words are non-standard words and need normalization, similar to previous work in [Han and Baldwin, 2011; Yang and Eisenstein, 2013; Li and Liu, 2014]. We can see from the table that: (1) The POS tagging accuracy of the system without normalization is worse than all the others with normalization. (2) In terms of normalization results, performance of the normalization system with a second-order Markov model is better than that using first-order. (3) Using joint decoding yields better accuracy for both normalization and POS tagging than the pipeline system that performs normalization and then POS tagging. (4) The joint decoding system with the normalization model trained from partially joint training with the perceptron strategy outperforms that with the models trained independently. (5) When all the non-standard tokens are correctly normalized (oracle setup), the POS tagging accuracy is the highest, as expected.

[4]This normalization result is the state-of-the-art performance on Test Set 2.

System	Test Set 1		Test Set 2	
	Norm	POS	Norm	POS
POS w/o Norm	0	90.22	0	90.54
Pipeline Norm† + POS	76.12	90.8	86.91	90.64
Pipeline Norm‡ + POS	76.96	90.91	87.01	90.68
Norm Oracle + POS	100	92.05	100	91.17
Joint decoding Separately Trained Model	77.03	91.04	87.15	90.72
Joint decoding Partially Joint Trained	**77.31**	**91.21**	**87.58**	**90.85**

Table 1: Normalization and POS tagging results from different systems on two data sets. All the results are accuracy (%). † Using first-order Markov Viterbi decoding in Norm system ‡ Using second order Markov model in Norm system.

5.3 Impact of Candidate Number

As mentioned in Section 4.2, the normalization candidate number is a key variable affecting the running time in our proposed method. Fortunately, a good normalization model can already rank the most possible candidates in the top. Using the token level normalization reranking results, we find that the top 20 candidates can already provide more than 90% precision in Test Set 2, though the top 1 accuracy is far from that; therefore it seems unnecessary to use more than 20 candidates in the joint decoding process. Figure 4 shows the average number of hidden states for each token when varying the maximum number of the normalization candidates. The Y-axis uses a relative value, in comparison with that when the candidate number is set as 1. We can see that on average the increasing rate is not bad (the worst case is N^3). In addition, a typical tweet rarely has more than 3 consecutive non-standard tokens. Table 2 shows the frequency of different numbers of consecutive non-standard tokens in the two test data sets. Obviously, most consecutive non-standard tokens are fewer than 3 tokens. The average consecutive non-standard token number in a tweet is 1.78 and 2.14 in the two data sets, while the average length of tweets in two test sets is 16.11 and 19.24 respectively. Therefore, the worst complexity we discussed earlier rarely happens in practice.

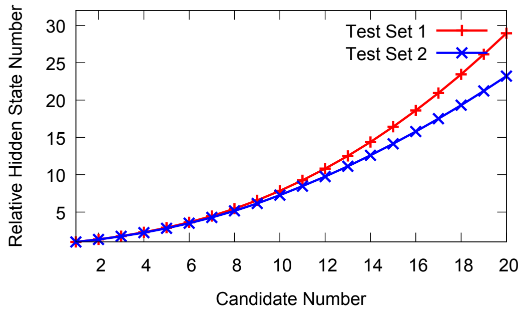

Figure 4: The average number of hidden states for each token in the two test sets when varying the number of normalization candidates.

# of consecutive non-standard token	1	2	3	4	5	6
Freq in Test Set1	1072	128	24	3	0	2
Freq in Test Set2	879	106	15	4	2	1

Table 2: Frequency of different numbers of consecutive non-standard tokens.

Intuitively deceasing the normalization candidate number for a non-standard token can speed up the decoding process, but hurts the normalization results and subsequently the POS tagging results. Therefore a study of the trade-off between the speed and accuracy is needed. Figure 5 shows the speed and performance as the number of candidates varies. The speed is also a relative value to that when the candidate number is 1. For example, in Test Set 1 when the candidate number is set as 20, its average speed of decoding a tweet is almost 70 times slower than that when candidate number is set as 1. From this Figure, we can see that when the candidate number is set to 5, both the normalization and POS tagging accuracy do not change much compared to when using 20 candidates, but the speed is about 4 times faster.

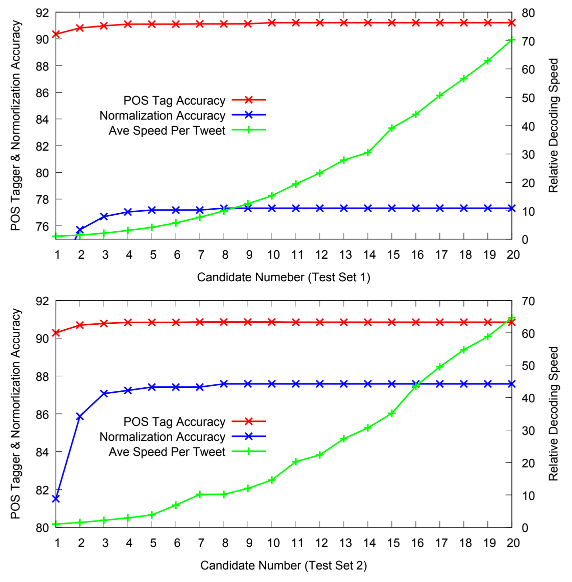

Figure 5: Tradeoff between performance and speed.

5.4 Error Analysis

Table 3 shows the POS tagging results separately for non-standard tokens and standard words on Test Set 1. We can see that normalization has a significant impact on the non-standard words. As expected, as normalization performance increases, the POS tagging errors for the non-standard words decrease. In comparison, the effect of normalization on the standard words is much smaller. In the oracle system, the POS tagging errors for the non-standard tokens are significantly lower than that of our proposed system, suggesting there is still quite some room for improvement for normalization and POS tagging.

System	Non-standard Token Err	Standard Word Err
POS w/o Norm	25.46	7.82
Pipeline Norm‡ + POS	19.90	7.65
Oracle	10.74	7.60
Joint decoding Partially Joint Trained	19.00	7.51

Table 3: POS tagging errors for non-standard tokens and standard words in Test Set 1. ‡ Using second order Markov model in Norm system.

A detailed error analysis further shows what improvement our proposed method makes and what errors it is still making. For example, for the tweet *'I can tell you tht my love ...'*, token *tht* is labeled as 'verb' when POS tagging is done on the original tweet (i.e., no normalization is performed). When POS tagging is applied to the normalized tweet, 'tht' is normalized as *that*, and the POS tag is also changed to the right one (subordinating conjunction). We noticed that joint decoding can solve some complicated cases that are hard for the pipeline system. Take the following tweet as an example: *'they're friiied after party !'*. Our joint decoding successfully corrects *friiied* to *fried* and thus labels *They're* as L(nominal+verbal). However, the pipeline system first corrected *friiied* as *friend*, and then POS tagging system labeled *They're* as D(determiner). In addition, when there are consecutive non-standard tokens, typically the joint decoding process tends to make a better decision. For example, *'tats crazin'* is part of a tweet, and its correct form is *'that's crazy'*. The pipeline system first normalizes *tats* to *thats* and *crazin* to *crazing*, and in the subsequent POS tagging step,'crazin' is labeled as noun, rather than adjective. But the joint decoding system correctly normalizes and labels both of them.

6 Conclusion and Further Work

In this paper we proposed a novel joint decoding approach to label every token's POS tag and correct the non-standard tokens in a tweet at the same time. This joint decoding combines information from the POS tagging model, the token level normalization scores, and the n-gram language model probabilities. Our experimental results demonstrate that normalization has a significant impact on POS tagging and our proposed method also improves both POS tagging and normalization accuracy and outperforms the previous work for both tasks. In addition, to our knowledge, we are the first to provide a tweet data set that contains both POS annotations and text normalization annotations for English corpus. Although our proposed method is more computationally expensive than the pipeline approach, it is applicable in practice when choosing suitable parameters, without performance loss. In the future, we plan to create a training set which has both POS tag and normalization annotation, allowing us to use the joint training strategy to train the POS tagging model and normalization model at the same time.

References

[Aw *et al.*, 2006] Aiti Aw, Min Zhang, Juan Xiao, and Jian Su. A phrase-based statistical model for sms text normalization. In *Processing of COLING/ACL*, 2006.

[Collins, 2002] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, 2002.

[Cook and Stevenson, 2009] Paul Cook and Suzanne Stevenson. An unsupervised model for text message normalization. In *Proceedings of NAACL*, 2009.

[Damerau, 1964] Fred J Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

[Foster *et al.*, 2011] Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, Josef Van Genabith, et al. # hardtoparse: Pos tagging and parsing the twitterverse. In *Proceedings of AAAI*, 2011.

[Gimpel *et al.*, 2011] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the ACL*, 2011.

[Han and Baldwin, 2011] Bo Han and Timothy Baldwin. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceeding of ACL*, 2011.

[Levenshtein, 1966] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

[Li and Liu, 2012a] Chen Li and Yang Liu. Improving text normalization using character-blocks based models and system combination. In *Proceedings of COLING*, 2012.

[Li and Liu, 2012b] Chen Li and Yang Liu. Normalization of text messages using character- and phone-based machine translation approaches. In *Proceedings of Interspeech*, 2012.

[Li and Liu, 2014] Chen Li and Yang Liu. Improving text normalization via unsupervised model and discriminative reranking. In *Proceedings of the ACL*, 2014.

[Liu *et al.*, 2011] Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. Recognizing named entities in tweets. In *Proceedings of ACL*, 2011.

[Liu *et al.*, 2012] Xiaohua Liu, Ming Zhou, Xiangyang Zhou, Zhongyang Fu, and Furu Wei. Joint inference of named entity recognition and normalization for tweets. In *Proceedings of ACL*, 2012.

[Martins and Smith, 2009] Andre F. T. Martins and Noah A. Smith. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the ACL Workshop on Integer Linear Programming for Natural Language Processing*, 2009.

[Owoputi *et al.*, 2013] Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*, 2013.

[Pennell and Liu, 2010] Deana Pennell and Yang Liu. Normalization of text messages for text-to-speech. In *ICASSP*, 2010.

[Pennell and Liu, 2011] Deana Pennell and Yang Liu. A character-level machine translation approach for normalization of sms abbreviations. In *Proceedings of IJCNLP*, 2011.

[Petrovic *et al.*, 2010] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. The edinburgh twitter corpus. In *Proceedings of NAACL*, 2010.

[Ritter *et al.*, 2010] Alan Ritter, Colin Cherry, and Bill Dolan. Unsupervised modeling of twitter conversations. In *Proceedings of the NAACL*, 2010.

[Ritter *et al.*, 2011] Alan Ritter, Sam Clark, Oren Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of EMNLP*, 2011.

[Yang and Eisenstein, 2013] Yi Yang and Jacob Eisenstein. A log-linear model for unsupervised text normalization. In *Proceedings of EMNLP*, 2013.

[Zhang and Clark, 2008] Yue Zhang and Stephen Clark. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL*, 2008.