Adapting a WSJ trained Part-of-Speech tagger to Noisy Text: Preliminary Results

Article · September 2011

DOI: 10.1145/2034617.2034623

Adapting a WSJ trained Part-of-Speech tagger to Noisy Text: Preliminary Results

Phani Gadde
Language Technologies Research Centre
IIIT-Hyderabad, India
phani.gadde@research.iiit.ac.in

L. V. Subramaniam
IBM Research
New Delhi, India
lvsubram@in.ibm.com

Tanveer A. Faruquie
IBM Research
New Delhi, India
ftanveer@in.ibm.com

ABSTRACT

With the increase in the number of people communicating through internet, there has been a steady increase in the amount of text available online. Most such text is different from the standard language, as people try to use various kinds of short forms for words to save time and effort. We call that noisy text. Part-Of-Speech (POS) tagging has reached high levels of accuracy enabling the use of automatic POS tags in various language processing tasks, however, tagging performance on noisy text degrades very fast. This paper is an attempt to adapt a state-of-the-art English POS tagger, which is trained on the Wall-Street-Journal (WSJ) corpus, to noisy text. We classify the noise in text into different types and evaluate the tagger with respect to each type of noise. The problem of tagging noisy text is attacked in two ways; a) Trying to overcome noise as a post processing step to the tagging b) Cleaning the noise and then doing tagging. We propose techniques to solve the problem in both the ways and critically compare them based on the error analysis. We demonstrate the working of the proposed models on a Short Message Service (SMS) dataset which achieve a significant improvement over the baseline accuracy of tagging noisy words by a state-of-the-art English POS tagger.

Keywords

POS tagging, noisy text, natural language processing, noisy text analytics, Part-of-Speech, adapting, NLP, robust towards noise

1. INTRODUCTION

Noise can be defined as any kind of difference in the surface form of an electronic text from the original, intended or actual text [16]. The distortion of words in short messaging service (SMS) and online forums like twitter, chat and discussion boards is essentially because the recipient can understand the message even if longer words are represented in short forms, thereby reducing the time and effort of the sender. Since not all internet users are native English speakers, we find foreign language words which are transliterated into English. This is another added complexity in online text.

Off-the-shelf language processing systems fail to work on noisy text due to several reasons, linguistic creativity being the major one. The very idea of seen and unseen words becomes totally different on these types of texts. For example, a word in the training data can appear in the test data in a different form. This means, each of the different words in the text may not be actually different, which makes the overall learning difficult. Considering the increase in the computer mediated communication (CMC) [15] across years, and the change in the language being used by the internet population, there is an urgent need to develop text processing systems that can deal with the noise in the text.

We attempt to do Part-Of-Speech (POS) tagging, the basic syntactic processing step in many high level text processing techniques like Named Entity Recognition (NER), Parsing and Question Answering (QA). A tagger for noisy text can be used in many practical applications on online text such as sentiment analysis on product reviews[3], online translation of text (emails, tweets) and information extraction [3, 2].

In this paper, the problem of tagging noisy text is attacked in two ways; a) Trying to overcome noise by correcting the tags of noisy words in a post processing step after tagging b) Cleaning the noise in the given sentence and then doing tagging. We propose techniques to solve the problem in both the ways and compare them based on the results and error analysis. We demonstrate the working of the proposed models on an SMS dataset which achieve a significant improvement over the baseline accuracy of tagging noisy words by Stanford POS tagger, which performs near state-of-the-art for English.

The rest of the paper is arranged as follows: Section 2 gives the related work on noisy text and parts-of-speech tagging. Section 3 explains the decisions taken before proposing the techniques, the data that is used and the tagger which is used as the baseline. The techniques are explained in section

4, followed by the experiments and results in section 5. We present a detailed error analysis in section 6 and conclude in section 7.

2. RELATED WORK

Part-Of-Speech (POS) tagging is a well studied problem in computational linguistics and natural language processing over the past few decades. This can be inferred from the state-of-the-art POS tagging accuracies not only of English, but also most of the other languages, which are near 95%, English being at 97.32%[23]. The high accuracy of trained POS taggers made them usable on-the-fly in many of the high level text processing techniques in natural language processing, information extraction and related areas.

POS taggers for English have been developed using several supervised learning techniques, most of which use graphical models like hidden markov models (HMM) [5], maximum entropy models (MEM)[22, 27, 26], conditional random fields (CRF)[17]. [8] and [23] use perceptron learning while the later used bidirectional sequence classification and achieved the state-of-the-art for the task. With different machine learning models reaching near state-of-the-art making the choice of the methods unimportant, handling unknown words has become the only prominent problem for English POS tagging in the last few years.

Many attempts were made to deal with the problem of tagging unknown words. [19] used a SVM based classifier framework to tag the unknown words in a post-processing step which can use the POS tags of known words in the data (context), making the prediction accurate. [29] uses various morphological features to find the correct tag of an unknown word, as there is a clear dependence between the tags and suffixes. In the problem of tagging noisy text, a large number of input noisy words would be unknown words as the tagger model which was obtained on a regular English corpus will not be able to correlate the short forms with their regular counterparts. However, the short forms are not completely unknown as their properties can be incurred from their regular counterparts once the words are related.

This work also finds its roots in spelling correction research, which assumes the input to be noisy. Recent spelling correction approaches have used statistical language models and stochastic finite state automata as a generative model for spelling errors [6]. Statistical models of pronunciation have also been used to improve such models [28]. [7] utilizes hand corrected training data consisting of original SMS and the corresponding expanded and corrected messages. From the hand generated parallel corpora of SMS and conventional language, a HMM is learned that mimics the SMS language generation. [24] investigated techniques for mapping non-standard words to standard but the focus of their technique was also on supervised learning. Their technique involved hand tagging of non-standard words with tags representing the type of corruption encountered and then learning to tag unknown words by the inferred type of corruption. Corrections were generated from the predicted tags. However, annotating non-standard words with their tags for the above two approaches is not straight forward, which we'll discuss in the later sections. [1] uses clustering to map non-standard words with their corresponding dictionary words using three character strings from the word as well as the context as features. This technique does not need annotated data.

[13] does parsing on a tree bank by adding noise to the data automatically. This work deals two types of noise, real word errors and un-grammatical sentences. [9] report interesting results on POS tagging of conversational speech. [4] does machine translation on texts having misspellings. They integrated the task of correcting the input text in the MT system using a character level confusion network. [12] is, to our knowledge, the first step towards evaluating the existing syntactic processing tools on noisy text. However, they don't give any automated methods to solve the problem of processing noisy text. [14] has annotated some tweets with a POS tagset developed for online texts [20] and has given preliminary tagging results using a CRF based model built on the annotated tagset. Instead of building a separate tagger for noisy text, we aim to adapt the existing state-of-the-art tagger for regular text to noisy text, with an intention of developing models that are robust towards these deviations of the text from the regular form.

3. ARRIVING AT THE PROBLEM

This section discusses various issues related to the data, approaches and goals in tagging noisy text.

3.1 Tagging noisy text

POS tagging for regular English has achieved reasonably high accuracy by using supervised learning techniques on large annotated datasets. But, we don't have a sufficiently large noisy data annotated with POS tags for training a model. Though the problem of tagging noisy text may seem to be a problem of not having training data at first, on a closer look at it, having a noisy training data is not the cure. For example, in one corpus we found *tomorrow* written in 28 different ways ranging from *tom*, *2moro*, *morrow*, to the actual. Creating a labeled dataset comprising of all the variants is impossible. A word in a noisy training data may occur in the test data in a different or distorted form. In such a case, that word is still a noisy word for a regular supervised tagger.

Another issue with creating noisy training data is that several forms of the same word appear sharing the context along with its distortions, increasing the sparsity. Hence, even if we have a noisy training data the learner requires some way of handling both sparsity while training and identifying the corresponding regular words while testing. Keeping this in mind, we attempt to adapt a state-of-the-art tagger trained on regular English (Wall Street Journal corpus [18]) (WSJ) text to noisy text, which leaves us the challenge of identifying the relatedness of different forms of a word while tagging, leaving the problem of handling sparsity in training for future work.

3.2 Evaluation

Due to the lack of standardization till now, evaluating a POS tagged noisy data also involves taking some important decisions. First of all, it is not trivial to annotate the noisy text, which is not only distorted at word level, but also at the structural level. For example, *I* and *am* get merged to form *im*, for which the decision of which POS tag to assign is tricky. [20] proposed a universal tagset suitable for annotating online texts. However, we don't annotate our data with these tags as we use a tagger trained on the WSJ.

On the other hand, our goal is not to build a machine learning system which works for the noisy text. The goal is to make the existing systems robust towards noise, i.e., developing system which can neglect the noise in the text, like humans. So, we decided to do a relative evaluation instead of manually annotating noisy text. We annotated each noisy word with the parallel regular word, whose output when given to a state-of-the-art POS tagger is considered as the gold-standard POS tag for that noisy word (like [10]). This is a fair assumption, considering the state-of-the-art accuracy of POS tagging on regular english text. Further, for merged words like *asap* described above, the output of the tagger is evaluated against the word that best denotes rhe meaning of the phrase (*soon* in *as soon as possible*).

3.3 Classifying the Noise

We first look into the nature of noise in CMC. For this we study a dataset of 847 SMEes from a publicly available SMS corpus used in [7]. This dataset contains a word aligned parallel corpus of noisy SMSes and their respective standard English forms. The SMS texts were manually translated to their standard English form and automatically aligned at the word level using a heuristic algorithm by the authors of [7]. The dataset contains only those text that were written completely in English (no code switching). Out of the 847 SMSes, 80% is used here as the development set, on which all the results in this paper are reported. There are around 18,000 tokens (words) in the translated standard text out of which around 2,000 are distinct. On average there are 80 characters in the SMS text for every 100 characters in the corresponding standard English text.

There are a lot of studies about different types of noise in CMC data in English, Arabic, German, Japanese and Swedish. The readers are referred to [7, 25] and the references therein, for the common trends in word compression and/or distortion.

After grouping similar kinds of noise and leaving some of the insignificant types, we classified the noise into four types. We used a simple algorithm to classify the noise of each noisy word based on comparison with the corresponding regular word. Table 1 below shows the four types of noise and examples of each type.

Table 1: Table showing different classifications of noise with examples.

Noise Type	Examples
Character Deletion	*yest* for *yesterday*
	ello for *hello*
	u for *you*
	hrdr for *harder*
Phonetic Substitution	*den* for *then*
	4 for *for*
	fink for *think*
	john for *John*
Abbreviations	*lol* for *laughing out loud*
	tb for *text back*
Dialectical Usage	*im* for *I am*
	gonna for *going to*

Character deletion corresponds to dropping of characters from the regular word whereas phonetic substitution may involve substitution of characters by similar sounding characters or digits. Abbreviations are the short forms in which each character corresponds to a word in the actual regular phrase. Dialectical usage is merging two words to form a new word based on the local usage of English. We distinguish dialectical usage from abbreviations as abbreviations are in general fixed and known to everyone whereas dialectical usage is a create word formation process which leads to an open class of different words, based on several factors like region and style (of language) of the SMS sender.

3.4 Tagger

We use the Stanford POS tagger [26], which is a maximum entropy based tagger. It models the sequence of words in a sentence as a bi-directional dependency network, which considers the lexical and tag context on both the sides to tag the current word. The tagger learns a log-linear conditional probability model from tagged text, using a maximum entropy method. Such a model is exponential with the parametric form:

$$p(t|h) = \frac{\prod_{j=1}^{K} \exp \lambda_j f_j(h,t)}{\sum_{t' \epsilon T} \prod_{j=1}^{K} \exp \lambda_j f_j(h,t')} \tag{1}$$

The model assigns a probability for every tag t in the tagset T given a word and its context h, which is usually defined as the sequence of several words and tags both preceding and succeeding the word, where $f_1 .... f_K$ are the features used while learning. The best tag sequence for a given word sequence is found using a variant of Viterbi algorithm. The feature functions f_j used for the best accuracy on regular English text are given in Table 2.

Table 2: Features used for baseline in Stanford POS tagger.

Feature	Description
BF	$< t_0, w_0 >, prefix, suffixes(w_0)$
TSF	$< t_0, t_{-1} >, < t_0, t_{-1}, t_{-2} >,$
	$< t_0, t_{-1}, t_{+1} >, < t_0, t_{+1} >$
	$< t_0, t_{+1}, t_{+2} >$
LF	$< t_0, w_{-1} >, < t_0, w_{+1} >$
MF	$< t_0, w_0, t_{-1} >, < t_0, w_0, t_{+1} >$
	$< t_0, w_{-1}, w_0 >, < t_0, w_0, w_{+1} >$
UWF	$isCapital(C), hasHyphen(H)$
	$hasNumber(N), conjunction(CHN)$
	$allCaps, isCapinContext$

BF: Basic Features
TSF: Tag Sequence Features
LF: Lexicalized Features
MF: Mixed Features
UWF: Unknown Word Features

Please refer to [27, 26] for the description of all the above features.

Note that the development data is not annotated with POS tags (as described in section 3.2). We evaluate the model's output by comparing it with the same model's output on the parallel regular text. This evaluation makes sense as our goal to develop systems which can neglect the noise in the text, like humans. In case of abbreviations and dialectical usage, the output tag is evaluated against the output of the dominant word in the group of words, which can possibly substitute the place of the abbreviation. Table 3 below shows the accuracy of tagging on the development set described in Section 3.1. Each row corresponds to a certain kind of noise. The second column gives the coverage of that kind of noise in the test set, while the third column gives the accuracy.

Table 3: Accuracy of the baseline tagger w.r.t. noise type along with the coverage

Noise Type	Coverage	Accuracy
Character Deletion	14.8	11.5
Phonetic Substitution	15.1	14.5
Abbreviations	2.2	5.4
Dialectical Usage	2.0	16.1
Noisy Words	34.2	12.74
Clean Words	65.8	91.2
Overall	100	64.45

Noisy Words in the first column of the second part of table 3 gives the combined accuracy of all the types of noisy words. *Clean Words* denotes all the words that are common to WSJ and the noisy dataset, which amount to around 65% of data. Note that, apart from a huge drop in the accuracy of noisy words (12.74) there is also considerable drop in the accuracy of clean words (from around 97% to 91.2%[1]). This can be attributed to the domain difference between the WSJ and the dataset and also to the noisy context while tagging the clean words. More analysis of the results is given in section 6. These results are provided here to show that there is a drastic drop in the accuracies when the noisy words percentage in the current dataset is around 35%.

4. TECHNIQUES

We consider two ways of dealing with the noise in tagging: a) Overcoming noise as a post processing step to tagging b) Cleaning the noise and then doing tagging. Section 4.1 introduces the technique in which the tagger's output is post-processed to deal with the noisy words whereas in section 4.2 we pre-process and clean the data before giving to the tagger.

4.1 Unknown word probability estimation (UWPE)

If we try to adapt a WSJ trained POS tagger to noisy text, the immediate consequence is the increase in the number of unknown words. Almost all the distorted words become unknown words for the tagger. Hence, we take an assumption that the unknown words for the tagger model are noisy words[2]. We modified the baseline tagger's code to output an additional cue *unk* for all the words which are not present in the tagger's model, so that the post-processing can be done only on those words. Also, as shown in Table 3, words which are not noisy are 65% in the development set and are tagged with an accuracy of 91.2%. On the other hand, noisy words are tagged with a mere 12.74% accuracy. Keeping this in mind, we intend to correct the tags of the unknown words for the tagger, in a post-processing step after tagging. We try to estimate the probability of giving a tag to a particular noisy word from the tags of the dictionary words which are possibly the noisy word's regular forms.

[1]Comparing 91.2% with the mean POS tagging accuracy of 97% is not fair, but it is discussed to show that the effect of noise is also on the clean words.

[2]*then* can be distorted to *den*, which is probably not an unknown word for the tagger. Currently, we are not handling this. However, we observed very few cases like this.

Figure 1: Noisy word *tnk* and its possible tags

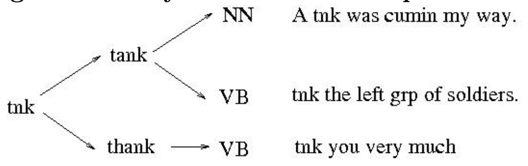
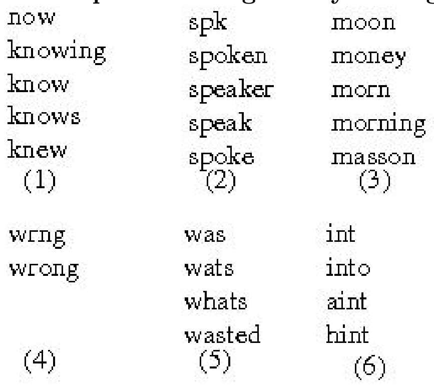


Figure 2: Sample clusters given by the algorithm



now	spk	moon
knowing	spoken	money
know	speaker	morn
knows	speak	morning
knew	spoke	masson
(1)	(2)	(3)
wrng	was	int
wrong	wats	into
	whats	aint
	wasted	hint
(4)	(5)	(6)

Figure 1 above explains this with the help of an example noisy word *tnk*. The idea is to predict the tag of word *tnk*, from the tags of word *tank* and *thank*, which are recognized to be its possible regular English forms.

We find the words which are the possible dictionary words for a noisy word by using a clustering that takes into account the noisy word as well as its context. We use the clustering technique proposed by [1] for non-standard words on the combined data of SMS and 0-18 sections of WSJ. We use a simplified HMM that is factorized in a way that leads to efficient training operated by clustering vectorial data instead of expensive dynamic programming. Clustering is based on cosine similarity between two words represented by the frequencies of sub-sequence strings from the word as well as the contextual words. As we can see in Table 3, most of the noise in our case is character deletion and phonetic substitution. Using this approach for the mapping model performs well because of the use of both sub-sequences in the noisy word as well as in a window of 4 on either side as features.

The output of the clustering is a predefined number of clusters having a mixture of noisy as well as dictionary words. Some example clusters are shown in Figure 2.

We use both the nearness between the noisy word and a dictionary word and the probability of assigning different tags to that dictionary word to estimate the probability of a specific tag becoming the correct tag of a specific noisy word. This can be seen in the following equation.

$$p(t_i|nw, h') = \frac{\sum_{j=1}^{C} p(t_i|rw_j, h') * p(rw_j|nw)}{\sum_{t_k \epsilon T} \sum_{j=1}^{C} p(t_k|rw_j, h') * p(rw_j|nw)} \qquad (2)$$

nw = current noisy word
rw_i = each regular (dictionary) word
h' = history excluding the current word
C = cluster size
T = POS tagset, t_i is a particular POS tag

where C is a set of dictionary words which are in the same cluster of the noisy word nw. In this way we introduce a random variable regular word (rw) between two random variables noisy word (nw) and POS tag (t). The denominator in the above equation is for normalization, which is the sum of numerator over all the tags. This can be ignored while tagging as what we actually need is maximum of all such emission probabilities, which is unaffected by the denominator. h' is used here instead of h for history to distinguish it from the history used in equation 1. Note that h' contains the tags of the dictionary words which are already predicted by the tagger. This probability estimation can be used given any model which can provide $p(t|rw, h')$ and any model which can provide $p(rw|nw)$. As mentioned earlier, we use Stanford tagger to get $p(t|rw, h')$. We use the Levenshtein's distance between the soundexes[3] of two words to get $p(rw|nw)$. We compute the soundex codes for each of the dictionary words in a cluster and the noisy word, and take the normalized edit distance of each dictionary word soundex with the noisy word, as shown in Equation 3 below. Each cluster may contain the synonyms of the expected dictionary words and also some misleading dictionary words. The use of soundex instead of just the lexical item is aimed to avoid all those misleading words in the same cluster, as the noisy words and the corresponding regular word sound similar.

$$p(rw|nw) = \frac{LD^{-1}(S(rw), S(nw))}{\sum_{rw' \epsilon C(nw)} LD^{-1}(S(rw'), S(nw))} \qquad (3)$$

where LD is the inverse Levenshtein's distance function, S is the soundex and $C(nw)$ is the set of dictionary words in the cluster of the noisy word nw. Since Levenshtein's distance is inversely correlated to the closeness between two words, the inverse of Levenshtein's distance is considered in equation 3.

This technique is similar to unsupervised tagging, where we cluster words in an unsupervised way and assign tags later. The working of the technique is demonstrated in Figure 3 below.

The dotted lines show the path which is less probable whereas the normal line shows the paths which are highly probable. In Figure 3a, when tnk is used for $tank$, if we use a mapping model which considers the context, $p(tank|tnk)$ will be higher. Also, any tagger model that considers the context should prefer NN instead of a VB thereby leading to the path $tnk \rightarrow tank \rightarrow NN$. It works in a similar way when tnk is used for $thank$ in Figure 3b. Since this is a post-processing technique, dictionary words which constitute around 65% of the words are already tagged and used as context while predicting the noisy word's tag.

[3]http://en.wikipedia.org/wiki/Soundex

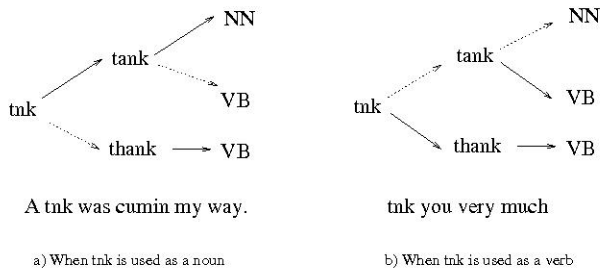


Figure 3: Probability estimation demo for the noisy word tnk

4.2 Tagging After Correction (TAC)

In this technique, we first come up with the best possible regular sentence RS for a given noisy sentence NS and then tag it using WSJ trained Stanford tagger. This is shown in the figure below.

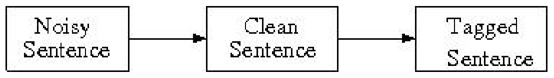


Figure 4: TAC flowchart

To obtain the regular sentence, we combine the clustering method discussed above with language modelling. We use the WSJ trained Stanford tagger model to find all the unknown words in a given sentence. As assumed in the UWPE technique, we consider them as noisy words and get the clusters by performing clustering on the combined data of SMS and 0-18 sections of WSJ, using the same technique [1] as in UWPE.

```
Input: Sentence $S$ with $n$ words
noisy words $k$
For each noisy word $nw$
$regularWords[nw] = regWordsFromCluster(nw);$

$Permutations\ RS =$
$allPermutations(S, regularWords)$

$bestProb = -1;$
$bestS = null;$
$for\ \ each\ \ sentence\ \ S'\ \ in\ \ RS:$

  $prob = \prod_{i=1}^{n} trigramProb(W_{i-2}, W_{i-1}, W_i)$
  $if\ \ prob\ >\ bestProb;$
    $bestProb = prob$
    $bestS = S'$

$StanfordTagger.tag(bestS)$
```

TAC Pseudocode

Once we get the possible regular words (by clustering) for each noisy word in the sentence, we generate all the permutations of possible regular sentences, substituting different combinations of regular words in place of the noisy words. We apply a trigram based language model trained on 0-18 sections of WSJ to assign probability based on the trigrams for each of those regular sentences and select the best one as the regular sentence for the given noisy sentence. This is shown in the above pseudocode.

regularWords is the array in which the regular words in the same cluster of the noisy word are stored. W_i in the *trigramProb*() function denotes the word at position i in the given sentence. We back-off to bigram probabilities and unigram probabilities in order to smooth the trigram probabilities obtained over 0-18 sections of WSJ. The features used for tagging in the Stanford tagger are those used for the baseline setup described in section 3.4.

5. EXPERIMENTS AND RESULTS

We use the dataset explained in Section 3.1 for the experiments. We divided the dataset of size 847 SMSes into development and test sets of 80% and 20% sizes respectively. Since we intend to do a lot more studies, all the experiments described in the paper are performed on the development set. Stanford tagger model with the feature setting explained in section 3.4 is considered as baseline for our experiments. Even in the UWPE and TAC techniques explained in sections 4.1 and 4.2 , the tagger model is the same as that used in the baseline. Differences between the techniques come from the way we deal with noise in both the techniques.

In the UWPE technique , experiments were done by varying the number of clusters from 500 to 2500, increasing 500 in each iteration. With small number of clusters, many unwanted words like different morphological variations of dictionary words and synonyms will fall into the cluster of a noisy word making the disambiguation between different dictionary words in the cluster difficult. On the other hand, if we use large number of clusters, the clusters miss even the relevant dictionary words for each noisy word. Our experiments with different number of clusters demonstrated this behavior clearly, giving the best performance at 1500 clusters.

Different soundex versions are also tried to get the word similarity in UWPE. Simple soundex[4] gives almost the same code if the first part of the words is the same (as in *work* and *worked*). Hence, to distinguish between different morphological variants of a dictionary word, we used several variants[5] of soundex like reverse soundex, DM soundex, metaphone and double metaphone. The best results were obtained when 1500 clusters and double metaphone [21] are used in UWPE.

Our primary goal in doing TAC is to see how the UWPE technique is able to tolerate noise by comparing it with a simple technique which opts for cleaning the noise before actually tagging it. The number of clusters parameter while doing TAC was also set to 1500 similar to UWPE. It is surprising to see TAC performing better than UWPE with a very simple formulation. More sophisticated techniques similar to [4] for English noisy to English regular text translation can be tried to correct the input before tagging.Table 4 below shows the accuracy comparison between the baseline, UWPE and TAC techniques.

[4]http://en.wikipedia.org/wiki/Soundex#Rules
[5]http://en.wikipedia.org/wiki/Soundex#Soundex_variants

Table 4: Accuracy comparison for Baseline, UWPE, and TAC

Noise Type	Baseline	UWPE	TAC
No Noise	91.2	90.7	91.3
Character Deletion	11.5	40.4	42.7
Phonetic Substitution	14.5	14.3	16.2
Dialectical Usage	16.1	8.6	9.4
Abbreviation	5.4	0.8	0.8
Overall	64.45	68.05	69
Overall Noisy Words	12.74	24.32	26.26

UWPE: Unknown Word Probability Estimation
TAC: Tagging After Correction

6. ANALYSIS

As we can see from Table 3, the overall baseline tagging accuracy is 64.45%, which is around 30% drop from the usual POS tagging accuracies. The difference between the overall accuracy and the accuracy of the noisy words (which is a mere 12.74%) is also quite high.

One more observation is that the accuracy of the words which are not noisy (regular words) dropped from 97.32% to 91%. There are at least three reasons for this decrease: a) The context of the regular words being noisy, b) Word and structure level distortion of SMS text, several function words are dropped in informal settings and c) the domain change between the WSJ corpus and the SMS text. At this point, we cannot make any comment on how much percentage of this drop is because of the noisy context, structure and how much is because of the change in the domain from WSJ.

Coming to the accuracy of the noisy words in the baseline, we did a further evaluation to know where actually the errors are. As expected, it turned out that most of the confusions are with the foreign word (FW) tag, which is preferred if a word is an unknown word for the tagger model and none of the rare word features (cf. [26]) help in tagging. Another major confusion is with the common noun (NN) tag, which is also expected, as the taggers are built to give nouns for unknown words (open class tags). Proper noun (NNP) has very low recall of 2.6%. This is because of the lack of capitalization for proper nouns in the noisy text.

As there would be a tendency of giving foreign word (FW) tag to most of the unknown words, the precision of FW tag is the lowest. Apart from NN, JJ and FW tags, which are given for most of the unknown words, precision of the tags is usually more than the recall, which means that the tagger is biased to a set of tags (NN, JJ, FW), since there are no explicit cues for the tagger to identify the tag. Conjuncts, prepositions, determiners and 'to' are tagged very efficiently, which might be because of the less distortion of those classes by virtue of their smaller length. Table 5 below shows a comparison for top 7 confusion pairs (sorted) in baseline with UWPE and TAC. It is evident that there is a consistent reduction in the confusions compared to baseline in case of UWPE and TAC, with TAC outperforming UWPE almost all the time. This phenomenon is more outstanding in case of present participles (VBG).

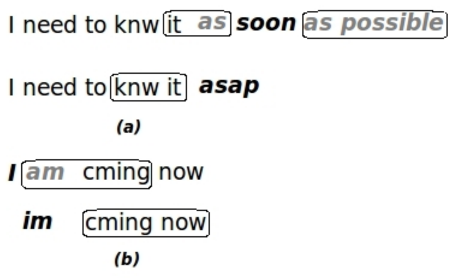
Note that the abbreviations and dialectic usages are evaluated w.r.t. the output of the dominant word in the expanded regular phrase. For example, *asap* which is the abbreviation of *as soon as possible* was evaluated against the output of the word *soon* which is the dominant of all, and can closely

Table 5: Top confusions between tags

Correct	Predicted	Baseline	UWPE	TAC
NN	FW	438	324	129
PRP	NN	436	101	100
PRP	FW	422	150	116
NNP	NN	327	344	440
NNP	FW	319	287	139
VBG	FW	266	237	25
VBP	NN	196	244	164

substitute *asap* in a sentence. The low accuracy for abbreviations and dialectical usage can be attributed to the loss of even the limited context that exists in case of the other kinds of noise. This can be seen in Figure 6. The noisy word and the corresponding word in the expanded phrase with which the output would be compared are in bold. The context is highlighted with a rectangle on the both sides of the words. The words in bold grey in the context in case of regular sentences are missing from their noisy counterparts because of merging. One can see that there is a clear difference in the contexts and one can't expect the tagger to perform similarly in a totally misleading context.

Figure 6. Context loss in abbreviations, dialectical usage.



The drop in the accuracies of abbreviations and dialectical usage below the baseline in case of both UWPE and TAC is because of the clustering used in both the approaches. Clustering is currently being done based on character differences of a noisy word with other regular words, which would group abbreviations and dialectical usages with other smaller regular words instead of their actual regular phrases (like *aint* in cluster (6)). Focused clustering that uses only the beginning letters of each word in a phrase will perform better in the case of dialectical usage while abbreviations should be dealt with a rule based post processor, as the list of abbreviations is fixed.

Compared to the abbreviations and dialectical usage, character deletion and phonetic substitution noise should be tagged better, as there is at least some context without loss. Though there is an improvement in the accuracy of character deletion noise, the accuracy of phonetic substitution noise was not improved significantly by any model. This can be either due to the bias in the clustering towards character deletion or due to the failure of soundex to capture the phonetic variation for SMS text. This is also observed in the clusters that are formed (can be seen in figure 2). They contain very few phonetic variants. Also, cluster (1) gives many morphological variations of a single dictionary word which ideally we want to eliminate using the soundex.

TAC, being simple and not very optimized, performed better than UWPE consistently across all the types of noise. It shows that the tagger is unable to tolerate the noise in the context while tagging text in UWPE. We observed a huge decrease in the confusion of tags with *FW* tag in case of TAC, as the tagger works on clean text. Having said that, it should be noted that the clean sentences obtained with TAC are not perfect most of the time. Often many function words are dropped which makes it difficult for the tagger.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we tried to adapt a state-of-the-art regular English trained tagger to noisy text, with a goal of making the model robust towards the noise. In that process, we compared two paradigms for tagging noisy text, i.e., correcting the tags of noisy text in a post-processor and tagging the text once it is corrected/cleaned in a pre-processing step. We proposed techniques for each of the paradigms and compared them on an SMS dataset. Both the approaches gave around 3.5% improvement in the overall tagging accuracy and nearly 13.52% improvement on the accuracy of noisy words. Direct tagging techniques fell short of TAC because of the tagger's inability to tolerate noise.

Phonetic substitution, which is a major part of the noise was not handled properly by any of the proposed techniques. The clustering model needs to be improved to distinguish the variations in phonetic substitution from character deletion, by using some kind of phonetic similarity.

The fact that the noisy sentences are unstructured is a problem that might be affecting the tagger as well as the language model that is being used in the methods proposed in this paper. Though this might be a potential reason for lower accuracy when the tagger is being adapted to noisy text, trying to make the tagger output those dropped words along with their tags is very hard. We would like to consider this in future. In this respect, like [11, 13], we intend to create artificial data (which mimics the natural noisy text) from regular data, which will also help us in doing better analysis as well as evaluation. We also intend to extend the ideas in this work to other types of noisy texts like tweets for which the annotation, features and basic results were explored by [14] using the universal POS tagset [20].

8. REFERENCES

[1] S. Acharyya, S. Negi, L. V. Subramaniam, and S. Roy. Language independent unsupervised learning of short message service dialect. *Int. J. Doc. Anal. Recognition*, (12):175–184, 2009.

[2] E. Bakshy, J. Hofman, W. Mason, and D. Watts. Everyone's an influencer: Quantifying influence on twitter. In *Proceedings of WSDM 2011*, pages 65–74. ACM, 2011.

[3] L. Barbosa and J. Feng. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of COLING 2010: Posters*, pages 36–44. Association for Computational Linguistics, 2010.

[4] N. Bertoldi, M. Cettolo, and M. Federico. Statistical machine translation of texts with misspelled words. In *Proceedings of HLT-NAACL*. Association for Computational Linguistics, 2010.

[5] T. Brants. Tnt – a statistical part-of-speech tagger. In *6th Applied Natural Language Processing Conference*, 2000.

[6] E. Brill and R. Moore. An improved model for noisy spelling correction. In *Proceedings of 38th Annual Meeting of the ACL*, pages 286–293, 2000.

[7] M. Choudhury, R. Sharaf, V. Jain, A. Mukherjee, S. Sarkar, and A. Basu. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*, 34:157–174, 2007.

[8] M. Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP*, 2002.

[9] V. Eidelman, Z. Huang, and M. Harper. Lessons learned in part-of-speech tagging of conversational speech. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 821–831. Association for Computational Linguistics, 2010.

[10] J. Foster. Parsing ungrammatical input: An evaluation procedure. In *Proceedings of LREC*, pages 2039–2042, 2004.

[11] J. Foster. Treebanks gone bad. *International Journal on Document Analysis and Recognition*, 10(3):129–145, 2007.

[12] J. Foster. cba to check the spelling investigating parser performance on discussion forum posts. In *HLT-NAACL, 2010*, pages 381–384. Association for Computational Linguistics, 2010.

[13] J. Foster, J. Wagner, and J. V. Genabith. Adapting a wsj-trained parser to grammatically noisy text. In *Proceedings of ACL-HLT, 2008*, pages 221–224. Association for Computational Linguistics, 2008.

[14] K. Gimpel, N. Schneider, B. O. Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of ACL-HLT*. Association for Computational Linguistics, 2011.

[15] S. C. Herring. Computer-mediated discourse. *Handbook of Discourse Analysis*, pages 612–634, 2001.

[16] C. Knoblock, D. Lopresti, S. Roy, and L. Subramaniam. Special issue on noisy text analytics. *International Journal on Document Analysis and Recognition*, 10(3):127–128, 2007.

[17] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *In The Eighteenth International Conference on Machine Learning*, 2001.

[18] M. Marcus, M. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[19] T. Nakagawa, T. Kudoh, and Y. Matsumoto. Unknown word guessing and part-of-speech tagging using support vector machines. In *Proceedings of the sixth natural language processing pacific rim symposium*, pages 325–331, 2001.

[20] S. Petrov, D. Das, and R. McDonald. A universal part-of-speech tagset. 2010.

[21] L. Philips. The double metaphone search algorithm., 2000.

[22] A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *EMNLP*, Philadelphia, PA, 1996.

[23] L. Shen, G. Satta, and A. Joshi. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 760–767, 2007.

[24] R. Sproat, A. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. Normalization of non-standard words. *Comput. Speech. Lang.*, 15:287–333, 1992.

[25] L. Subramaniam, S. Roy, T. Faruquie, and S. Negi. A survey of types of text noise and techniques to handle noisy text. In *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, pages 115–122. ACM, 2009.

[26] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the AAAI'94 Workshop on CaseBased Reasoning*, pages 252–259, 2003.

[27] K. Toutanova and C. D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP*, 2000.

[28] K. Toutanova and R. Moore. Pronunciation modelling for improved spelling correction. In *Proceedings of 40th AnnualMeeting of the ACL*, pages 144–151, 2002.

[29] H. Tseng, D. Jurafsky, and C. Manning. Morphological features help pos tagging of unknown words across language varieties. In *Proceedings of the Fourth SIGHAN workshop on Chinese language processing*, pages 32–39, 2005.