Normalizing Microtext

Zhenzhen Xue, Dawei Yin and Brian D. Davison
Department of Computer Science & Engineering, Lehigh University
Bethlehem, PA 18015 USA
{zhx208, day207, davison} @cse.lehigh.edu

Abstract

The use of computer mediated communication has resulted in a new form of written text—*Microtext*—which is very different from well-written text. Tweets and SMS messages, which have limited length and may contain misspellings, slang, or abbreviations, are two typical examples of microtext. Microtext poses new challenges to standard natural language processing tools which are usually designed for well-written text. The objective of this work is to normalize microtext, in order to produce text that could be suitable for further treatment.

We propose a normalization approach based on the source channel model, which incorporates four factors, namely an orthographic factor, a phonetic factor, a contextual factor and acronym expansion. Experiments show that our approach can normalize Twitter messages reasonably well, and it outperforms existing algorithms on a public SMS data set.

Introduction

The Web has become the channel in which people communicate, write about their lives and interests, and give opinions and rate products. In some contexts, users, especially young users, write in an informal way without minding spelling or grammar, even deliberately shortening the words or using slang. Words or phrases such as "*lol*" (*laugh out loud*), "*c u 2nite*" (*see you tonight*) and "*plz*" (*please*) which may not be found in standard English are widely used by Web users. The casual usage of language results in a new form of written text which is very different from well-written text. This chat-speak-style text is especially prevalent in Short Message Service (SMS), chat rooms and micro-blogs. Such chat-speak-style text is referred to as *Microtext* by (Ellen 2011). In this work, Tweets and SMS messages are explored as typical examples of microtext.

There has been a big effort to produce natural language processing (NLP) algorithms and tools that try to understand well-written text, but these tools cannot be applied out of the box to analyze microtext which usually contains noise, including misspellings, abbreviations and improper grammar. Tools such as Named Entity Recognition have been reported to have substantially lower performance on Tweets than on structured text, partly due to the high amount of noise present in Tweets (Murnane 2010; Corvey et al. 2010). In order to effectively retrieve or mine data from microtext, it is necessary to normalize the microtext, so that they are more readable for machines and humans, and become suitable for further treatment using standard NLP tools.

The task of microtext normalization addressed in this work has many similarities to traditional spelling correction but also poses additional challenges. Both the frequency and severity of spelling errors in microtext are significantly greater than in normal text. In our datasets, a significant fraction of microtext (31% of Twitter messages, 92% of SMS messages) requires normalization. In microtext, sometimes words are misspelled due to typographic errors such as "*positon*" for "*position*". In other cases, such *Non-Standard Words* (NSWs), words which are written differently from their standard forms, are used intentionally for various reasons, including phonetic spelling (*enuf* for *enough*), emotional emphasis (*goooood* for *good*), popular acronyms (*asap* for *as soon as possible*), etc. Some NSWs are so widely used such that they are more recognizable than their original standard forms such as *ATM*, *CD*, *PhD* and *BMW*. Typically, the validity of a microtext term cannot be decided by lexicon look-up or by checking its grammaticality. Therefore, general purpose spelling correction methods which rely on lexicon look-up are not able to solve the problem of normalizing microtext.

In this paper, we address the problem of normalizing microtext using the source channel model. The problem of normalizing microtext is viewed as that of recovering the intended word or word sequence given an observed word which may or may not be a NSW. Our model will take four factors into consideration, namely, character string-based typographical similarity, phonetic similarity, contextual similarity and popular acronyms.

Related Work

The following sections compare microtext normalization with similar and related applications.

General Text Normalization

General text normalization has been well studied in text-to-speech (Sproat et al. 2001). General text normalization deals with tokens such as numbers, abbreviations, dates, currency amounts and acronyms which are normative while microtext

normalization needs to deal with lingo such as "*l8*" (*late*) and "*msg*" (*message*), which are typically self-created and are not yet formalized in linguistics. Moreover, microtext is brief, containing a very limited number of characters (140 characters for Tweets, and 160 characters for SMS messages). It is thus much more difficult to use contextual or grammatical information.

Spelling Correction

Short message normalization has many similarities with traditional spelling correction which has a long history. There are two branches of research in conventional spelling correction, which deal with non-word errors and real-word errors respectively.

Non-word correction is focused on generating and ranking a list of possible spelling corrections for each word not found in a spelling lexicon. The ranking process usually adopt some lexical-similarity measure between the misspelled string and the candidates considering different edit operations (Damerau 1964; Levenshtein 1966), or a probabilistic estimation of the likelihood of the correction candidates (Brill and Moore 2000; Toutanova and Moore 2002). Real-word spelling correction is also referred to as context-sensitive spelling correction, which tries to detect incorrect usage of valid words in certain contexts (Golding and Roth 1996; Mangu and Brill 1997).

Spelling correction algorithms rely on spelling lexicons to detect misspelled tokens, which is an unrealistic approach for microtext normalization. Short messages often contain valid words that are not found in any traditional lexicon (e.g., *iTune, Picasa*), sometimes may even contain in-lexicon words that are actually intended to be other legitimate words (*I don't no* for *I don't know*, and *agree wit u* for *agree with you*). In the open Web, we clearly cannot construct a static trusted lexicon, as many new names and concepts become popular every day and it would be difficult to maintain a high-coverage lexicon. Therefore, the models used in spelling correction are inadequate for microtext normalization.

SMS Normalization

While there are few attempts to normalize Twitter messages or microtext in general, there is some work on SMS normalization. Text messages, which are also called Short Message Service (SMS) texts, are similar to Tweets considering the length of messages and the devices used to generate the messages.

Text message normalization has been handled through three well-known NLP metaphors: spelling correction, machine translation and automatic speech recognition. The spelling correction metaphor (Choudhury et al. 2007; Cook and Stevenson 2009) performs the normalization task on a word-per-word basis by applying the noisy channel model. The machine translation metaphor, first proposed by Aw et al. (2006), considers the process of normalizing text messages as a statistical phrase-based machine translation task from a source language (the text message) to a target language (its standard written form). Kobus et al. (2008) proposed to handle text message normalization through an automatic speech recognition metaphor based on the observation that text messages present a lot of phonetic spellings. Beaufort et al. (2010) took a hybrid approach which combines spelling correction and machine translation methods.

Our generic approach, detailed below, incorporates components from both spelling correction and automatic speech recognition.

Normalization Model

Given a piece of microtext, our model will normalize terms one by one. Thus, the challenge is to determine the corresponding standard form t, which can be a term or a sequence of terms, for each observed term t'. Notice that t may or may not be the same with t'. t' is a NSW if $t' \neq t$. Thus, the task is to find the most probable normalization t^* for each observed term t':

$$t^* = \arg\max_t P(t|t') = \arg\max_t P(t'|t)P(t) \qquad (1)$$

$P(t'|t)$ models the noisy channel through which an intended term t is sent and is corrupted into the observed term t'. $P(t)$ models the source that generates the intended term t. In practice, the source model can be approximated with an n-gram statistical language model.

Since a non-standard term and its corresponding standard form might be similar from one or multiple perspectives(s), it is reasonable to assume that there are several channels, each of which would distort an intended term in one of the above aspects. More specifically, a grapheme channel would be responsible for the spelling distortion, a phoneme channel would cause phonetic corruptions, a context channel may change terms around a target term and an acronym channel may shrink a sequence of terms into one term. In reality, an intended word may be transferred through one or multiple channels, and an observed term might be a mixed corruption from more than one channel. Under this assumption, and letting $\{c_k|c_k \in C\}$ denote the set of channels, Equation 1 can be further developed to

$$\begin{aligned} t^* &= \arg\max_t \sum_k P(t', c_k|t)P(t) \\ &= \arg\max_t \sum_k P(t'|c_k, t)P(c_k|t)P(t) \end{aligned} \qquad (2)$$

A term t is transferred through channel c_k with probability $P(c_k|t)$. Within the channel c_k, term t is distorted to term t' according to the channel model $P(t'|c_k, t)$.

Channel Models

In this section, we will describe and formulate each of the four channels in detail.

Grapheme Channel. The grapheme channel models the distortion of spellings. One method to model this channel is to consider $P(t'|c_{grapheme}, t)$ to be proportional to the similarity between a NSW and its normalization. The more similar a normalization candidate is to the NSW t', the more likely it is the correct normalization for t'. This gives

$$P(t'|c_{grapheme}, t) \propto Sim(t', t)$$

t'	**Candidates**	**Contexts & *Normalizations***
yr	your, year	it's yr (*your*) turn happy new yr (*year*)
wll	will, well, wall	i wll (*will*) leave
bin	bin, been	have bin (*been*) really happy since ...
no	no, know	don't no (*know*) why

Table 1: The importance of context in normalization

where $Sim(t', t)$ is the string-based similarity measure which is evolved from (Kothari et al. 2009):

$$Sim(t', t) = \frac{LCSRatio(t', t)}{Distance(CS(t'), CS(t))} \quad (3)$$

In Equation 3, $LCSRatio(t', t)$ means *Longest Common Subsequence Ratio* between t' and t, which is the ratio of the length of their LCS and the length of the longer string. $Distance(CS(t'), CS(t))$ is the weighted *Consonant Skeleton Levenshtein Distance* between t' and t which gives lower weight to deletion operations based on the observation that NSWs often delete characters from, but seldomly add characters to their standard forms. For example, a NSW "*abt*" is more likely to mean "*about*" than "*at*". The consonant skeleton of a term is generated by first removing consecutive repeated characters and then removing all vowels.

Phoneme Channel. The phoneme channel is responsible for distortion in pronunciations. Similar to the grapheme channel, the probability of t being transformed into t' is proportional to the similarity between the two terms, except that the similarity is measured on the phonetic representations instead of on orthographic forms:

$$P(t'|c_{phoneme}, t) \propto Sim(L2P(t'), L2P(t))$$

Thus, a crucial step is Letter-to-Phoneme (L2P) conversion, which is to estimate the pronunciation of a term, represented as a sequence of phonemes, from its orthographic form, represented as a sequence of letters. There has been a lot of research on letter-to-phoneme conversion including works from van den Bosch et al. (2006), Jiampojamarn et al. (2008), Bartlett et al. (2008), and Rama et al. (2009). We use the method proposed by Rama et al. which treats the L2P problem as a machine translation problem and has the translation performance comparable to the state-of-art algorithms such as Jiampojamarn et al. Our L2P model is trained based on the CMU Pronouncing Dictionary (CMUDict) using the optimal parameters specified by Rama et al. (2009).

After the L2P conversion, the similarity measure between two phoneme sequences is the same as the similarity measure used in the grapheme channel except that a uniform weight Levenshtein distance is used instead of weighted Levenshtein distance.

Context Channel. Contextual information provides useful clues to find the most probable normalization. Table 1 lists several examples in which context plays an important role during normalization. In each example, the table shows the observed NSW (t'), selected normalization candidates, the context in which the NSW appears and the correct normalization. An ambiguous NSW, such as the example of "*yr*", may refer to different standard words in different contexts. With the help of context, the correct normalization may be determined by considering the n-gram probability. There are also some situations in which real-word errors need to be handled, such as the examples "*bin*" and "*no*" in Table 1.

Our context channel model is based on an n-gram language model, which is trained on a large Web corpus and would return a probability score for a query word or phrase. The context channel model can be written as

$$P(t'|c_{context}, t) \propto prob(Ctx_t)$$

where $prob()$ is the probability score based on the language model, and Ctx_t is the context (sequence of terms) containing the normalization candidate t. Ctx_t consists of up to N words to either side of the target word, where $N = 3$. Thus, Ctx_t could be trigram, bigram or unigram (when no context is available).

For each normalization candidate t_{ij} of term t'_i, there is a different context containing the candidate itself. In general, when the current term has preceding terms, we would use its normalized preceding terms as its context. Otherwise, the following $N - 1$ non-normalized terms would be used. The preceding terms are preferred to succeeding terms because of the fact that a term being examined is usually not surrounded by clean context. Based on the assumption that the normalization for all or at least most of the preceding terms are correct, using normalized preceding terms provides more reliable context information than using non-normalized succeeding terms.

Acronym Channel. All of the above three channel models deal with word-to-word normalization. However, there are popular acronyms such as "*asap*", "*fyi*" and "*lol*" which are widely used and involve word-to-phrase mappings. The acronym channel is a complementary model which takes into consideration the one-to-many mapping.

$$P(t'|c_{acronym}, t) = \begin{cases} 1, & \text{if } (t', t) \in A \\ 0, & \text{otherwise} \end{cases}$$

$A = \{(t', t)|t' \text{ is the acronym for } t\}$ is a set of popular acronyms and their corresponding full forms collected from a public website[1]. Thus, if a normalization candidate is a full form for the observed term, it would get a score of 1.

Channel Probabilities

In Equation 2, channel models are combined using channel probabilities $P(c_k|t)$.

Generic Channel Probabilities. For the reason of simplicity, an assumption that word t and channel c_k are inde-

[1] `http://www.webopedia.com/quick_ref/textmessageabbreviations.asp`

pendent of each other is made and Equation 2 can be simplified to

$$t^* = \arg\max_t \sum_k P(t'|c_k, t)P(c_k|t)P(t)$$
$$\simeq \arg\max_t \sum_k P(t'|c_k, t)P(c_k)P(t)$$

in which $P(t)$ is the language model, $P(c_k)$ is the prior probability for channel c_k, and $P(t'|c_k, t)$ is the channel model.

The probability of a term being emitted through a certain channel is independent of the term itself. In addition, the probabilities of the four channels can be tuned to achieve the best normalization performance.

Term Dependent Channel Probabilities. The assumption that term t is independent of a noisy channel c_k may not always be true. Some terms are more likely to be written using phonetic spellings, while others tend to be abbreviated based on orthographic similarities. Ideally, the model needs to estimate $P(c_k|t)$, term dependent channel probabilities. Thus, $P(c_k|t)$, the probability of a term t being transferred through a specific channel c_k needs to be learned from training data.

Given a term t' with J normalization candidates, and each candidate t_j with a score G_j from the grapheme channel, score P_j from the phoneme channel, score C_j from the context channel and score A_j from the acronym channel, function f is defined as

$$f_j = \alpha G_j + \beta P_j + \gamma C_j + \delta A_j$$

where α, β, γ and δ are term-dependent channel probabilities.

Suppose that t_l is the correct normalization. The system would output a correct normalization if $f_l \geq f_j (1 \leq j \leq J)$ holds. Naturally, the goal is to learn the value of α, β, γ and δ for t_l such that $\sum_{j=1}^{J} f_l - f_j$ is maximized. Thus, the objective function h is defined as

$$h = \arg\max \sum_{j=1}^{J} s(f_l - f_j)$$

where $s(x)$ is the sigmoid function.

The optimal combination weights can be achieved by taking the gradient descent method in which the four parameters are updated in each step until they converges.

By taking this strategy, the probabilities of the four channels are estimated per term. There would be a group of optimal channel probabilities (α, β, γ, and δ) for each correction normalization.

For each term t' in a short message, the normalization candidates are terms in CMUDict with the same initial letter and t' itself. Candidates are then ranked according to Equation 2. The candidate which is ranked highest would be the output of normalization.

Experiments

Experiment Setup

In order to evaluate the normalization performance of our system, we did experiments on two different datasets. One is a Twitter message dataset. A public SMS dataset is additionally used to compare our performance against previous work.

A Twitter message is a typical microtext which has limited length. For the normalization task, we manually normalized 818 tweets which were collected in September 2010. Among these tweets, 253 (31%) messages need normalization, with a total of 479 word transformations. A word following @ or # would not be normalized as we consider these two symbols as signifying reserved tokens (user id or hashtag) in tweets.

SMS messages, which are similar to Twitter messages, can be viewed as another type of microtext. Our experiments are based on a public parallel English SMS dataset provided by Choudhury et al. (2007). The training set has 854 text messages and the test set has 138 text messages. In the test set, 127 (92%) messages need normalization, with an average of 4 word transformations per message.

We considered two kinds of language models in our experiments. One is the Microsoft Web N-gram Service[2] which is a public Web service that provides statistical language models based on Web data (Wang et al. 2010). The other is a language model built from 18,000,000 raw tweets, which includes 3,872,108 unigrams, 27,093,464 bigrams and 10,010,590 trigrams. While there is no significant difference between the experiment results by using the two different language models on the Twitter dataset, Web N-gram language model performs better than Twitter language model on the SMS dataset. The following results are reported based on the Microsoft Web N-gram language model.

Two variations of our model, multi-channel model with generic channel probabilities (MC-Generic) and multi-channel model with term-dependent channel probabilities (MC-TD), are compared with the current best result (Choudhury et al. 2007) on the public SMS dataset. We also compare with two other approaches: the standard spell checker Aspell, and a machine translation method (MT) represented by the Moses (Koehn et al. 2007) toolkit with the factored model. Both MC-TD and MT approaches require training data. Thus, on the Twitter dataset, the results of the MT method and MC-TD model are reported based on 5-fold cross validation. The SMS dataset comes with training data, which is used by these two approaches.

We consider following four standard metrics in the evaluation process, namely, accuracy, precision, recall and F-measure.

Results

Table 2 compares the performance of our system with that of baseline algorithms on the two datasets. The SMS dataset is relatively small and clean compared with the Twitter dataset. Thus, all algorithms achieve better performance on this dataset.

Overall, the performance of Aspell is far behind the other algorithms on both datasets. While the Moses toolkit performs better than the spelling correction method proposed in (Choudhury et al. 2007), our model outperforms all the

[2] `http://Web-ngram.research.microsoft.com`

	Twitter Dataset				SMS Dataset				
	Aspell	Moses	MC-Generic	MC-TD	Aspell	Choudhury07	Moses	MC-Generic	MC-TD
Accuracy	0.92	0.94	**0.96**	**0.96**	0.63	0.86	0.92	0.95	**0.96**
F-measure	0.06	0.41	**0.61**	**0.61**	0.13	n/a	0.87	0.89	**0.91**
Precision	0.08	**0.76**	0.50	0.50	0.23	n/a	0.88	0.89	**0.93**
Recall	0.05	0.28	0.78	**0.79**	0.09	n/a	0.86	**0.90**	**0.90**

Table 2: Normalization results on Twitter and SMS dataset.

comparison algorithms by having higher accuracy and F-measure. When breaking F-measure into precision and recall, notice that our multi-channel model achieves much higher recall than the comparison algorithms, especially on Twitter, which means that our model is able to identify more NSWs. This is because that our system actually models the correspondences between NSWs and their normalizations while the MT method learns the correspondences by rote. For example, word "*love*" is observed in several different but similar non-standard forms including "*luv*", "*lurve*" and "*lurrrrvve*". All these three NSWs can be properly handled by our multi-channel model. But in the MT method, learning one of these non-standard forms doesn't help in predicting the correct normalization for the other two. Thus, the advantage of our model is that it captures the lexical creativity observed in microtext, which enables the system to normalize new NSWs. Twitter messages contain a lot of entity names and people names which increase the difficulty of normalization. Our model has a relatively low precision on the Twitter dataset because it normalizes some entity names wrongly. Since the MT method only normalizes NSWs that are learned during the training phase, it can achieve higher precision than our model.

MC-TD model performs slightly better than MC-Generic model on the SMS dataset, while the two models have roughly the same performance on the Twitter dataset. In MC-Generic model, a correct normalization candidate may not be ranked the first among a set of candidates due to suboptimal channel weights. The MC-TD model has the potential to improve the performance by providing a more accurate estimation of channel probabilities, which would reduce such ranking errors. However, it is expensive to estimate the term-dependent weights in MC-TD model, requiring a large amount of data. From the experiments, the improvement of MC-TD over MC-Generic model is not significant considering the computational cost of MC-TD.

We further analyze the usefulness of each channel in the MC-Generic model. Since the acronym channel maps from words to phrases, which is different from the other three channels, it is always necessary to include the acronym channel. Thus, we only investigate the weights of the other three channels. The ternary plots in Figure 1 show the influence of channel weights on system performance, in which each dimension represents one of the three channels, and performance scores are grouped into classes in which larger numbers correspond to better performance. The Figure suggests that the three channels are contributing to the performance to different extents. In the Twitter message dataset, the grapheme and phoneme channels have comparable contributions to the overall performance, while the context channel has a relatively small contribution. The best normalization performance is achieved when the channel probabilities are $\alpha = 0.25$, $\beta = 0.2$, $\gamma = 0.05$ and $\delta = 0.5$ where α, β, γ and δ correspond to the grapheme channel, phoneme channel, context channel and acronym channel respectively. In the SMS dataset, the best performance is achieved when the weights of the three channels are roughly balanced ($\alpha = 0.15$, $\beta = 0.15$, $\gamma = 0.2$ and $\delta = 0.5$). Using the grapheme channel alone can get reasonable performance, while adding the phoneme and context channels help to improve the normalization result. Compared with the performance from Twitter messages, the context channel plays a more important role in the SMS dataset.

Conclusions and Future Work

This paper proposed a multi-channel model inspired by source channel theory to normalize microtext. Our model outperforms baseline algorithms on two typical datasets. We showed that orthographic, phonetic and contextual information and acronym expansion are four important factors for the normalization of microtext.

We note that a paper to be published shortly by Han and Baldwin (2011) uses some of the same intuitions as our work, but on different data and with an evaluation approach that assumes perfect recognition of ill-formed words. We hope to compare our approach to their work in the near future. In addition, we want to investigate the impact of microtext normalization on other NLP or IR tasks.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Number IIS-0916152.

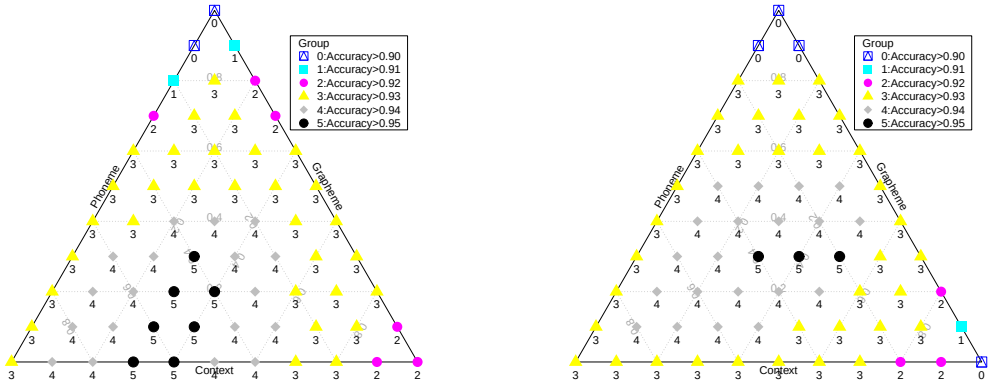
References

Aw, A.; Zhang, M.; Xiao, J.; and Su, J. 2006. A phrase-based statistical model for SMS text normalization. In *Proc. of the COLING/ACL on Main conference poster sessions*, 33–40.

Bartlett, S.; Kondrak, G.; and Cherry, C. 2008. Automatic syllabification with structured SVMs for letter-to-phoneme conversion. In *Proc. of ACL-08: HLT*, 568–576.

Beaufort, R.; Roekhaut, S.; Cougnon, L.-A.; and Fairon, C. 2010. A hybrid rule/model-based finite-state framework for normalizing sms messages. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, 770–779.

Brill, E., and Moore, R. C. 2000. An improved error model for noisy channel spelling correction. In *Proc. of the 38th Annual*



(a) Twitter dataset

(b) SMS dataset

Figure 1: Influence of channel weights on normalization performance.

Meeting on Association for Computational Linguistics, ACL '00, 286–293.

Choudhury, M.; Saraf, R.; Jain, V.; Mukherjee, A.; Sarkar, S.; and Basu, A. 2007. Investigation and modeling of the structure of texting language. *Int. J. Doc. Anal. Recognit.* 10:157–174.

Cook, P., and Stevenson, S. 2009. An unsupervised model for text message normalization. In *Proc. of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, 71–78.

Corvey, W. J.; Vieweg, S.; Rood, T.; and Palmer, M. 2010. Twitter in mass emergency: what NLP techniques can contribute. In *Proc. of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, WSA '10, 23–24.

Damerau, F. J. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM* 7:171–176.

Ellen, J. 2011. All about microtext: A working definition and a survey of current microtext research within artificial intelligence and natural language processing. In *Proc. of the Third International Conference on Agents and Artificial Intelligence*.

Golding, A., and Roth, D. 1996. Applying winnow to context-sensitive spelling correction. 182–190.

Han, B., and Baldwin, T. 2011. Lexical normalisation of short text messages: Makn sens a # twitter. In *Proc. of the 49th Annual Meeting on Association for Computational Linguistics*, ACL '11. To appear.

Jiampojamarn, S.; Cherry, C.; and Kondrak, G. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proc. ACL*, 905–913.

Kobus, C.; Yvon, F.; and Damnati, G. 2008. Normalizing SMS: are two metaphors better than one? In *Proc. of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, 441–448.

Koehn, P.; Hoang, H.; Birch, A.; Callison-Burch, C.; Federico, M.; Bertoldi, N.; Cowan, B.; Shen, W.; Moran, C.; Zens, R.; Dyer, C.; Bojar, O.; Constantin, A.; and Herbst, E. 2007. Moses: open source toolkit for statistical machine translation. In *Proc. of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, 177–180.

Kothari, G.; Negi, S.; Faruquie, T. A.; Chakaravarthy, V. T.; and Subramaniam, L. V. 2009. SMS based interface for FAQ retrieval. In *Proc. of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL-IJCNLP '09, 852–860.

Levenshtein, V. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 10:707.

Mangu, L., and Brill, E. 1997. Automatic rule acquisition for spelling correction. In *Proc. of the 14th International Conference on Machine Learning*, 734–741.

Murnane, W. 2010. Improving Accuracy of Named Entity Recognition on Social Media Data. Master's thesis, University of Maryland.

Rama, T.; Singh, A. K.; and Kolachina, S. 2009. Modeling letter-to-phoneme conversion as a phrase based statistical machine translation problem with minimum error rate training. In *NAACL '09: Proc. of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, 90–95.

Sproat, R.; Black, A. W.; Chen, S.; Kumar, S.; Ostendorf, M.; and Richards, C. 2001. Normalization of non-standard words. *Computer Speech and Language* 15(3):287–333.

Toutanova, K., and Moore, R. C. 2002. Pronunciation modeling for improved spelling correction. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL '02, 144–151.

van den Bosch, A., and Canisius, S. 2006. Improved morpho-phonological sequence processing with constraint satisfaction inference. In *Proc. of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology and Morphology*, SIGPHON '06, 41–49.

Wang, K.; Thrasher, C.; Viegas, E.; Li, X.; and Hsu, B.-j. P. 2010. An overview of Microsoft web N-gram corpus and applications. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, HLT-DEMO '10, 45–48. Stroudsburg, PA, USA: Association for Computational Linguistics.