Unsupervised cleansing of noisy text

Danish Contractor
IBM India Software Labs
dcontrac@in.ibm.com

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

L. Venkata Subramaniam
IBM Research India
lvsubram@in.ibm.com

Abstract

In this paper we look at the problem of cleansing noisy text using a statistical machine translation model. Noisy text is produced in informal communications such as Short Message Service (SMS), Twitter and chat. A typical Statistical Machine Translation system is trained on parallel text comprising noisy and clean sentences. In this paper we propose an unsupervised method for the translation of noisy text to clean text. Our method has two steps. For a given noisy sentence, a weighted list of possible clean tokens for each noisy token are obtained. The clean sentence is then obtained by maximizing the product of the weighted lists and the language model scores.

1 Introduction

Noisy unstructured text data is found in informal settings such as Short Message Service (SMS), online chat, email, social message boards, newsgroup postings, blogs, wikis and web pages. Such text may contain spelling errors, abbreviations, non-standard terminology, missing punctuation, misleading case information, as well as false starts, repetitions, and special characters.

We define noise in text as any kind of difference between the surface form of a coded representation of the text and the correct text. The SMS "u kno whn is d last train of delhi metro" is noisy because several of the words are not spelled correctly and there are grammar mistakes. Obviously the person who wrote this message intended to write exactly what is there in the SMS. But still it is considered noisy because the message is coded using non-standard spellings and grammar.

Current statistical machine translation (SMT) systems rely on large parallel and monolingual training corpora to produce high quality translations (Brown et al., 1993). Most of the large parallel corpora available comprise newswire data that include well formed sentences. Even when web sources are used to train a SMT system, noisy portions of the corpora are eliminated (Imamura et al., 2003) (Imamura and Sumita, 2002) (Khadivi and Ney, 2005). This is because it is known that noise in parallel corpora results in incorrect training of models thus degrading the performance.

We are not aware of sufficiently large parallel datasets comprising noisy and clean sentences. In fact, even dictionaries comprising of noisy to clean mappings in one language are very limited in size.

With the increase in noisy text data generated in various social communication media, cleansing of such text has become necessary. The lack of noisy parallel datasets means that this problem cannot be tackled in the traditional SMT way, where translation models are learned based on the parallel dataset. Consider the problem of translating a noisy English sentence e to a clean English sentence h. SMT imagines that e was originally conceived in clean English which when transmitted over the noisy channel got corrupted and became a noisy English sentence. The objective of SMT is to recover the original clean sentence.

Coling 2008: Poster Volume, pages 189–196,
Beijing, August 2010

The goal of this paper is to analyze how noise can be tackled. We present techniques to translate noisy text sentences e to clean text sentences h. We show that it is possible to clean noisy text in an unsupervised fashion by incorporating steps to construct ranked lists of possible clean English tokens and then searching for the best clean sentence. Of course as we will show for a given noisy sentence, several clean sentences are possible. We exploit the statistical machine learning paradigm to let the decoder pick the best alternative from these possible clean options to give the final translation for a given noisy sentence.

The rest of the paper is organized as follows. In section 2 we state our contributions and give an overview of our approach. In Section 3 we describe the theory behind clean noisy text using MT. In Section 4 we explain how we use a weighing function and a plain text dictionary of clean tokens to guess possible clean English language tokens. Section 5 describes our system along with our results. We have given an analysis of the kind of noise present in our data set in section 5.2

2 Our Approach

In this paper we describe an unsupervised method to clean noisy text. We formulate the text cleansing problem in the machine translation framework using translation model 1 (Brown et al., 1993). We clean the text using a pseudo-translation model of clean and noisy words along with a language model trained using a large monolingual corpus. We use a decoder to search for the best clean sentence for a noisy sentence using these models.

We generate scores for the pseudo translation model using a weighing function for each token in an SMS and use these scores along with language model probabilities to hypothesize the best clean sentence for a given noisy SMS. Our approach can be summarized in the following steps:

- Tokenize noisy SMS S into n tokens s_1, s_2 ... s_n. For each SMS token s_i create a weighted list based on a weighing function. These lists along with their scores corresponds to the translation probabilities of the SMT translation model.
- Use the lists generated in the step above along with clean text language model scores, in a decoder to hypothesize the best clean sentence
- At the end of the search choose the highest scoring sentence as the clean translation of the noisy sentence

In the above approach we do not learn the translation model but emulate the translation model during decoding by analyzing the noise of the tokens in the input sentence.

3 Noisy sentence translation

Statistical Translation models were invented by Brown, et al (Brown et al., 1993) and are based on the source-channel paradigm of communication theory. Consider the problem of translating a noisy sentence e to a clean sentence h. We imagine that e was originally conceived cleanly which when transmitted over the noisy communication channel got corrupted and became a noisy sentence. The goal is to get back the original clean sentence from the noisy sentence. This can be expressed mathematically as

$$\hat{h} = arg\max_{h} Pr(h|e)$$

By Bayes' Theorem

$$\hat{h} = arg\max_{h} Pr(e|h)Pr(h)$$

Conceptually, the probability distribution $P(e|h)$ is a table which associates a probability score with every possible pair of clean and noisy sentences (e, h). Every noisy sentence e is a candidate translation of a given clean sentence h. The goodness of the translation $h \Rightarrow e$ is given by the probability score of the pair (e, h). Similarly, $Pr(h)$ is a table which associates a probability score with every possible clean sentence h and measures how well formed the sentence h is.

It is impractical to construct these tables exactly by examining individual sentences (and sentence pairs) since the number of conceivable sentences in any language is countably infinite. Therefore, the challenge in Statistical Machine Translation is to construct approximations to the probability

distributions $P(e|h)$ and $Pr(h)$ that give an acceptable quality of translation. In the next section we describe a model which is used to approximate $P(e|h)$.

3.1 IBM Translation Model 2

IBM translation model 2 is a generative model, i.e., it describes how a noisy sentence e could be stochastically generated given a clean sentence h. It works as follows:

- Given a clean sentence h of length l, choose the length (m) for the noisy sentence from a distribution $\epsilon(m|l)$.
- For each position $j = 1, 2, \ldots m$ in the noisy string, choose a position a_j in the clean string from a distribution $a(a_j|j, l, m)$. The mapping $\mathbf{a} = (a_1, a_2, \ldots, a_m)$ is known as alignment between the noisy sentence e and the clean sentence h. An alignment between e and h tells which word of e is the corrupted version of the corresponding word of h.
- For each $j = 1, 2, \ldots m$ in the noisy string, choose an noisy word e_j according to the distribution $t(e_j|h_{a_j})$.

It follows from the generative model that probability of generating $e = e_1 e_2 \ldots e_m$ given $h = h_1 h_2 \ldots h_l$ with alignment $\mathbf{a} = (a_1, a_2, \ldots, a_m)$ is

$$Pr(e, a|h) = \epsilon(m|l) \prod_{j=1}^{m} t(e_j|h_{a_j}) a(a_j|j, m, l).$$

It can be easily seen that a sentence e could be produced from h employing many alignments and therefore, the probability of generating e given h is the sum of the probabilities of generating e given h under all possible alignments a, i.e., $Pr(e|h) = \sum_a Pr(e, a|h)$. Therefore,

$$Pr(e|h) =$$

$$\epsilon(m|l) \sum_{a_1=0}^{l} .. \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(e_j|h_{a_j}) a(a_j|j, m, l).$$

The above expression can be rewritten as follows:

$$Pr(e|h) = \epsilon(m|l) \prod_{j=1}^{m} \sum_{i=0}^{l} t(e_j|h_i) a(i|j, m, l).$$

Typical statistical machine translation systems use large parallel corpora to learn the translation probabilities (Brown et al., 1993). Traditionally such corpora have consisted of news articles and other well written articles. Therefore in theory $P(e|h)$ should be constructed by examining sentence pairs of clean and noisy sentences. There exists some work to remove noise from SMS (Choudhury et al., 2007) (Byun et al., 2008) (Aw et al., 2006) (Neef et al., 2007) (Kobus et al., 2008). However, all of these techniques require an aligned corpus of SMS and conventional language for training.

Aligned parallel corpora for noisy sentence is difficult to obtain. This lack of data for a language and the domain dependence of noise makes it impractical to construct corpus from which $P(e|h)$ can be learnt automatically. This leads to difficulty in learning $P(e|h)$. Fortunately the alignment between clean and noisy sentences are monotonic in nature hence we assume a uniform distribution for $a(i|j, m, l)$ held fixed at $(l+1)^{-1}$. This is equivalent to model 1 of IBM translation model. The translation models $t(e_j|h_{a_j})$ can be thought of as a ranked list of noisy words given a clean word. In section 4.2 we show how this ranked list can be constructed in an unsupervised fashion.

3.2 Language Model

The problem of estimating the sentence formation distribution $Pr(h)$ is known as the language modeling problem. The language modeling problem is well studied in literature particularly in the context of speech recognition. Typically, the probability of a n-word sentence $h = h_1 h_2 \ldots h_n$ is modeled as $Pr(h) = Pr(h_1|H_1)Pr(h_2|H_2) \ldots Pr(h_n|H_n)$, where H_i is the *history* of the ith word h_i. One of the most popular language models is the n-gram model (Brown et al., 1993) where the history of a word consists o f the word and the previous $n-1$ words in the sentence, i.e., $H_i = h_i h_{i-1} \ldots h_{i-n+1}$. In our application we use a smoothed trigram model.

3.3 Decoding

The problem of searching for a sentence h which minimizes the product of translation model prob-

ability and the language model probability is known as the decoding problem. The decoding problem has been proved to be NP-complete even when the translation model is IBM model 1 and the language model is bi-gram (K Knight., 1999). Effective suboptimal search schemes have been proposed (F. Jelinek, 1969), (C. Tillman et al., 1997).

4 Pseudo Translation Model

In order to be able to exploit the SMT paradigm we first construct a pseudo translation model. The first step in this direction is to create noisy token to clean token mapping. In order to process the noisy input we first have to map noisy tokens in noisy sentence, S^e, to the possible correct lexical representations. We use a similarity measure to map the noisy tokens to their clean lexical representations .

4.1 Similarity Measure

For a term $t_e \in \mathcal{D}^e$, where $\mathcal{D}^e$ is a dictionary of possible clean tokens, and token s_i of the noisy input S^e, the similarity measure $\gamma(t_e, s_i)$ between them is

$$\gamma(t_e, s_i) = \begin{cases} \frac{LCSRatio(t_e,s_i)}{EditDistance_{SMS}(t_e,s_i)} & \text{if } t_e \text{ and } s_i \text{ share same starting character} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $LCSRatio(t_e, s_i) = \frac{length(LCS(t_e,s_i))}{length(t_e)}$ and $LCS(t_e, s_i)$ is the *Longest common subsequence* between t_e and s_i. The intuition behind this measure is that people typically type the first few characters of a word in an SMS correctly. This way we limit the possible variants for a particular noisy token.

The *Longest Common Subsequence Ratio* (LCSRatio) (Melamed et al., 1999) of two strings is the ratio of the length of their LCS and the length of the longer string. Since in the SMS scenario, the dictionary term will always be longer than the SMS token, the denominator of LCSR is taken as the length of the dictionary term.

The $EditDistance_{SMS}$ (Figure 1) compares the Consonant Skeletons (Prochasson et al., 2007) of the dictionary term and the SMS token. If the Levenshtein distance between consonant skeletons is small then $\gamma(t_e, s_i)$ will be high. The intuition behind using $EditDistance_{SMS}$ can be explained through an example. Consider an SMS token "gud" whose most likely correct form is "good". The two dictionary terms "good" and "guided" have the same LCSRatio of 0.5 w.r.t "gud", but the $EditDistance_{SMS}$ of "good" is 1 which is less than that of "guided", which has $EditDistance_{SMS}$ of 2 w.r.t "gud". As a result the similarity measure between "gud" and "good" will be higher than that of "gud" and "guided". Higher the *LCSRatio* and lower the $EditDistance_{SMS}$, higher will be the similarity measure. Hence, for a given SMS token "byk", the similarity measure of word "bike" is higher than that of "break".

In the next section we show how we use this similarity measure to construct ranked lists. Ranked lists of clean tokens have also been used in FAQ retrieval based on noisy queries (Kothari et al., 2009).

Procedure $EditDistance_{SMS}(t_e, s_i)$
Begin
  return $LevenshteinDistance(CS(s_i), CS(t_e)) + 1$
End

Procedure *CS (t): // Consonant Skeleton Generation*
Begin
  Step 1. remove consecutive repeated characters in t
    // $(fall \rightarrow fal)$
  Step 2. remove all vowels in t
    //$(painting \rightarrow pntng, threat \rightarrow thrt)$
  return t
End

Figure 1: $EditDistance_{SMS}$

4.2 List Creation

For a given noisy input string S^e, we tokenize it on white space and replace any occurrence of digits to their string based form (e.g. 4get, 2day) to get a series of n tokens $s_1, s_2, \ldots, s_n$. A list L_i^e is created for each token s_i using terms in a dic-

hv u cmplted ure prj rprt
d ddline fr sbmission of d rprt hs bn xtnded
i wil be lte by 20 mns
d docs shd rech u in 2 days
thnk u for cmg 2 d prty

Figure 2: Sample SMS queries

tionary D^e consisting of clean english words. A term t_e from D^e is included in L_i^e if it satisfies the threshold condition

$$\gamma(t_e, s_i) > \phi \quad (2)$$

Heuristics are applied to boost scores of some words based on positional properties of characters in noisy and clean tokens. The scores of the following types of tokens are boosted:

1. Tokens that are a substring of a dictionary words from the first character.

2. Tokens having the same first and last character as a dictionary word.

3. Token that are dictionary words themselves (clean text).

The threshold value ϕ is determined experimentally. Thus we select only the top scoring possible clean language tokens to construct the sentence.

Once the list are constructed the similarity measure along with the language model scores is used by the decoding algorithm to find the best possible English sentence. It is to be noted that these lists are constructed at decoding time since they depend on the noisy surface forms of words in the input sentence.

5 Experiments

To evaluate our system we used a set of 800 noisy English SMSes sourced from the publicly available National University of Singapore SMS corpus[1] and a collection of SMSes available from the Indian Institute of Technology, Kharagpur. The SMSes are a collection of day-to-day SMS exchanges between different users. We manually

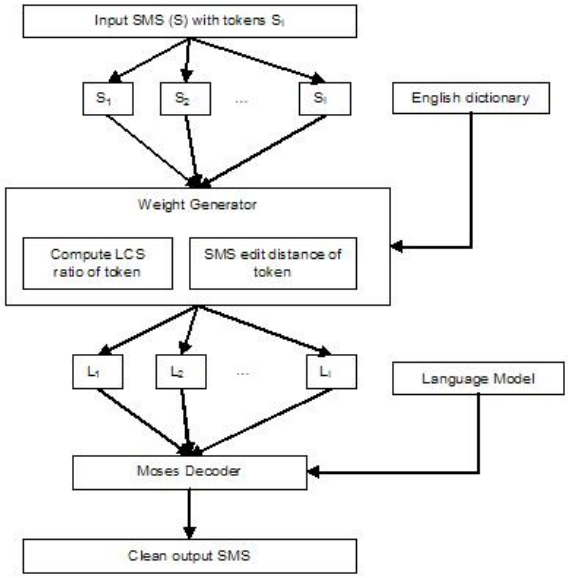

Figure 3: System implementation

	BLEU scores	1-gram	2-gram	3-gram	4-gram
Noisy text	40.96	63.7	45.1	34.5	28.3
Cleaned text	53.90	77.5	58.7	47.4	39.5

Table 1: BLEU scores

generated a cleaned english version of our test set to use as a reference.

The noisy SMS tokens were used to generate clean text candidates as described in section 4.2. The dictionary D^e used for our experiments was a plain text list of 25,000 English words. We created a tri-gram language model using a collection of 100,000 clean text documents. The documents were a collection of articles on news, sporting events, literature, history etc. For decoding we used Moses[2], which is an open source decoder for SMT (Hoang et al., 2008), (Koehn et al., 2007). The noisy SMS along with clean candidate token lists, for each SMS token and language model probabilities were used by Moses to hypothesize the best clean english output for a given noisy SMS. The language model and translation models weights used by Moses during the decoding phase, were adjusted manually after some experimentation.

We used BLEU (Bilingual evaluation understudy) and Word error rate (WER) to evaluate the performance of our system. BLEU is used to

[1] http://wing.comp.nus.edu.sg/downloads/smsCorpus

[2] http://www.statmt.org/moses/

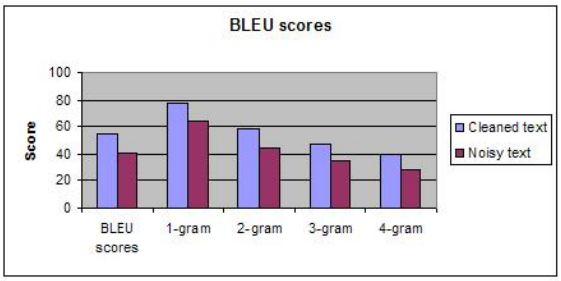

Figure 4: Comparison of BLEU scores

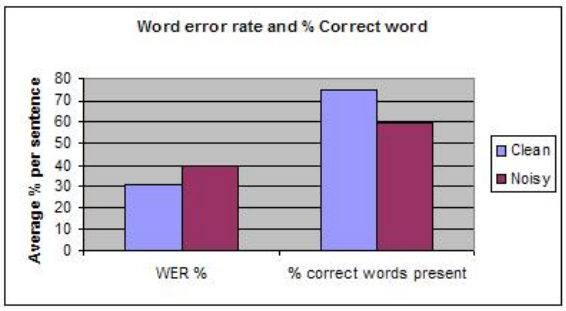

Figure 5: Word error rates

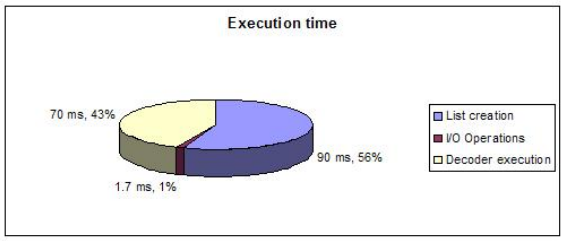

Figure 6: Execution time slices

establish similarity between a system translated and human generated reference text. A noisy SMS ideally has only one possible clean translation and all human evaluators are likely to provide the same translation. Thus, BLEU which makes use of n-gram comparisons between reference and system generated text, is very useful to measure the accuracy of our system. As shown in Fig 4 , our system reported significantly higher BLEU scores than unprocessed noisy text.

The word error rate is defined as

$$WER = \frac{S + D + I}{N} \quad (3)$$

where S is the number of substitutions, D is the number of the deletions, I is the number of the insertions and N is the number of words in the reference The WER can be thought of as an execution of the Levenstein Edit distance algorithm at the token level instead of character level.

Fig 5 shows a comparison of the WER. Sentences generated from our system had 10 % lower WER as compared to the unprocessed noisy sentences. In addition, the sentences generated by our system match a higher number of tokens (words) with the reference sentences, as compared to the noisy sentences.

5.1 System performance

Unlike standard MT system when $P(e|h)$ is precomputed during the training time, list generation in our system is dynamic because it depends on the noisy words present in the input sentence. In this section we evaluate the computation time for list generation along with the decoding time for finding the best list. We used an Intel Core 2 Duo 2.2 GHz processor with 3 GB DDR2 RAM to implement our system. As shown in Fig 6 the additional computation involving list creation etc takes up 56% (90 milliseconds) of total translation time. 43% of the total execution time is taken by the decoder, while I/O operations take only 1% of the total execution time. The decoder execution time slices reported above exclude the time taken to load the language model. Moses took approximately 10 seconds to load our language model.

5.2 Measuring noise level in SMS queries

The noise in the collected SMS corpus can be categorized as follows

1. Removal of characters : The commonly observed patterns include deletion of vowels (as in “msg” for “message”), deletion of repeated character (as in ”happy” for “hapy”) and truncation (as in “tue” for “tuesday”)

Type of Noise	% of Total Noisy Tokens
Deletion of Characters	48%
Phonetic Substitution	33%
Abbreviations	5%
Dialectical Usage	4%
Deletion of Words	1.2%

Table 2: Measure of Types of SMS Noise

	Clean (Reference) text	Noisy text	Output text
Perplexity	19.61	34.56	21.77

Table 3: Perplexity for Reference, Noisy Cleaned SMS

2. Phonetic substitution: For example, “2” for “to” or “too”, “lyf”’ for “life”, “lite” for “light” etc.

3. Abbreviation: Some frequently used abbreviations are “tb” for “text back”, “lol” for “laughs out loud”, “AFAICT” for “as far as i can tell” etc.

4. Dialectal and informal usage: Often multiple words are combined into a single token following certain dialectal conventions. For example, “gonna” is used for “going to”, “aint” is used for “are not”, etc.

5. Deletion of words: Function words (e.g. articles) and pronouns are commonly deleted. “I am reading the book” for example may be typed as “readin bk”.

Table 2 lists statistics on these noise types from 101 SMSes selected at random from our data set. The average length of these SMSes was 13 words. Out of the total number of words in the SMSes, 52% were non standard words. Table 2 lists the statistics for the types of noise present in these non standard words.

Measuring character level perplexity can be another way of estimating noise in the SMS language.The perplexity of a LM on a corpus gives an indication of the average number of bits needed per n-gram to encode the corpus. Noise results in the introduction of many previously unseen n-grams in the corpus. Higher number of bits are needed to encode these improbable n-grams which results in increased perplexity.

We built a character-level language model (LM) using a document collection (vocabulary size is 20K) and computed the perplexity of the language model on the noisy and the cleaned SMS test-set and the SMS reference data.

From Table 3 we can see the difference in perplexity for noisy and clean SMS data. Large perplexity values for the SMS dataset indicates a high level of noise. The perplexity evaluation indicates that our method is able to remove noise from the input queries as given by the perplexity and is close to the human correct reference corpus whose perplexity is 19.61.

6 Conclusion

We have presented an inexpensive, unsupervised method to clean noisy text. It does not require the use of a noisy to clean language parallel corpus for training. We show how a simple weighing function based on observed heuristics and a vocabulary file can be used to shortlist clean tokens. These tokens and their weights are used along with language model scores, by a decoder to select the best clean language sentence.

References

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*.

Jeunghyun Byun, Seung-Wook Lee, Young-In Song, Hae-Chang Rim. 2008. Two Phase Model for SMS Text Messages Refinement. *In Proceedings of AAAI Workshop on Enhanced Messaging*.

Aiti Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. *In Proceedings of COLING-ACL.*

Guimier de Neef, Emilie, Arnaud Debeurme, and Jungyeul Park. 2007. TILT correcteur de SMS : Evaluation et bilan quantitatif. *In Actes de TALN*, Toulouse, France.

Catherine Kobus, Francois Yvon and Geraldine Damnati. 2008. Normalizing SMS: Are two metaphors better than one? *In Proceedings of COLING*, Manchester.

Sreangsu Acharya, Sumit Negi, L Venkata Subramaniam, Shourya Roy. 2009. Language independent unsupervised learning of short message service dialect. *International Journal on Document Analysis and Recognition*.

Philipp Koehn, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, Evan Herbst 2007. Moses: Open source toolkit for statistical machine

translation. *In Proceedings of ACL, Demonstration Session* .

Peter F. Brown, Vincent J.Della Pietra, Stephen A. Della Pietra, Robert. L. Mercer 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation *Computational Linguistics*.

I. D. Melamed. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics.*

E. Prochasson, C. Viard-Gaudin, and E. Morin. 2007. Language models for handwritten short message services. *In Proceedings of ICDAR.*

S. Khadivi and H. Ney. 2005. Automatic filtering of bilingual corpora for statistical machine translation. *In Proceedings of NLDB*, pages 263–274, 2005.

K. Imamura and E. Sumita. 2002. Bilingual corpus cleaning focusing on translation literality. In *In Proceedings of ICSLP*.

K. Imamura, E. Sumita, and Y. Matsumoto. 2003. Automatic construction of machine translation knowledge using translation literalness. In *In Proceedings of EACL.*

K. Knight, 1999. Decoding complexity in word replacement translation models. *Computational Linguistics*.

F. Jelinek, 1969. A fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development*.

C. Tillman, S. Vogel, H. Ney, and A. Zubiaga. 1997. A DP-based search using monotone alignments in statistical translation. *In Proceedings of ACL.*

Hieu Hoang, Philipp Koehn. 2008. Design of the Moses decoder for statistical machine translation. *In Proceedings of ACL Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing*.

Govind Kothari, Sumit Negi, Tanveer A. Faruquie, Venkatesan T. Chakraverthy, L. Venkata Subramaniam. 2009. SMS based interface for FAQ retrieval, In *In Proceedings of ACL-IJCNLP*