# Spell Checking Techniques for Replacement of Unknown Words and Data Cleaning for Haitian Creole SMS Translation

**Sara Stymne**
Department of Computer and Information Science
Linköping University, Sweden
sara.stymne@liu.se

## Abstract

We report results on translation of SMS messages from Haitian Creole to English. We show improvements by applying spell checking techniques to unknown words and creating a lattice with the best known spelling equivalents. We also used a small cleaned corpus to train a cleaning model that we applied to the noisy corpora.

## 1 Introduction

In this paper we report results on the WMT 2011 featured shared task on translation of SMS messages from Haitian Creole into English, which featured a number of challenges. The in-domain data available is small and noisy, with a lot of non-standard language. Furthermore, Haitian Creole is a low resource language, for which there are few language technology tools and corpora available.

Our main focus has been to make the best possible use of the available training data through different ways of cleaning the data, and by replacing unknown words in the test data by plausible spelling equivalents. We have also investigated effects of different ways to combine the available data in translation and language models.

## 2 Baseline system

We performed all our experiments using a standard phrase-based statistical machine translation (PBSMT) system, trained using the Moses toolkit (Koehn et al., 2007), with SRILM (Stolcke, 2002) and KenLM (Heafield, 2011) for language modeling, and GIZA++ (Och and Ney, 2003) for word alignment. We also used a lexicalized reordering model (Koehn et al., 2005). We optimized each system separately using minimum error rate training (Och, 2003). The development and devtest data were available in two versions, as *raw*, noisy data, and in a *clean* version, where the raw data had been cleaned by human post-editors.

The different subcorpora had different tokenizations and casing conventions. We normalized punctuation by applying a tokenizer that separated most punctuation marks into separate tokens, excluding apostrophes that were suspected to belong to contracted words or Haitian short forms, periods for abbreviations, and periods in URLs. There were often many consecutive punctuation marks; these were replaced by only the first of the punctuation marks. In the English translations of the SMS data there were often translator's notes at the end of the translations. These were removed when introduced by two standard formulations: *Additional Notes* or *translator's note/interpretation*. In addition the translation marker *The SMS [ . . . ]* were removed.

Case information was inconsistent, especially for SMS data, and for this reason we lower-cased all Haitian source data. On the English target side we wanted to use true-cased data, since we wanted case distinctions in the translation output. We based the true-casing on Koehn and Haddow (2009), who changed the case of the first word in each sentence, to the most common case variant of that word in the corpus when it is not sentence initial. In the noisy SMS data, though, there were many sentences with all capital letters that would influence this truecasing method negatively. To address this, we modified the algorithm to exclude sentences with more than 40% capital letters when calculating corpus statistics, and to lowercase all unknown capitalized words.



*Proceedings of the 6th Workshop on Statistical Machine Translation*, pages 470–477,
Edinburgh, Scotland, UK, July 30–31, 2011. 

| Data | Sentences | Words | TM | LM | Reo | TC |
|---|---|---|---|---|---|---|
| In-domain SMS data | 17,192 | 35k | SMS | SMS | yes | yes |
| Medical domain | 1,619 | 10k | other | other | – | – |
| Newswire domain | 13,517 | 30k | other | other | – | yes |
| Glossary | 35,728 | 85k | other | other | – | – |
| Wikipedia parallel sentence | 8,476 | 90k | other | other | – | yes |
| Wikipedia named entities | 10,499 | 25k | other | other | – | – |
| Haitisurf dictionary | 1,687 | 3k | other | other | – | yes |
| Krengle sentences | 658 | 3k | other | other | – | yes |
| The Bible | 30,715 | 850k | bible | bible | – | yes |

Table 1: Corpora used for training translation models (TM), language models (LM), lexicalized reordering model (Reo), and true-casing model (TC). All corpora are bilingual English–Haitian Creole.

All translation results are reported for the devtest corpus, on truecased data. We report results on three metrics, Bleu (Papineni et al., 2002), NIST (Doddington, 2002), and Meteor optimized on fluency/adequacy (Lavie and Agarwal, 2007).

## 3 Corpus Usage

The corpora available for the task was a small bilingual in-domain corpus of SMT data, a limited amount of bilingual out-of-domain corpora, such as dictionaries and the Bible. This is different to the common situation of domain adaptation, as in the standard WMT shared tasks, where there is a small bilingual in-domain corpus, a larger in-domain monolingual corpus, and possibly several out-of-domain corpora that can be both monolingual and bilingual. In such a scenario it is often useful to use all available training data for both translation and language models, possibly in separate models (Koehn and Schroeder, 2007).

Table 1 summarizes how we used the available corpora, in our different models. For translation and language models we separated the bilingual data into three parts, the SMS data, the Bible, and everything else. For our lexicalized reordering model we only used SMS data, since we believe word order there is likely to differ from the other corpora. For the English true-casing model we concatenated the English side of all bilingual corpora that were not lower-cased.

Table 2 shows the results of the different model combinations on the clean devtest data. When we used only the SMS data in the translation model, the scores changed only slightly regardless of which combinations of language models we used. Using two translation models for the SMS data and the other bilingual data overall gave better results than when only using SMS data for the translation model. With double translation models it was best only to use the SMS data in the language model. Including the Bible data had a minor impact. Based on these experiments we will use all available training data in two translation models, one for SMS and one for everything else, but only use SMS data in one language model, which corresponds to the line marked in bold in Table 2, and which we will call the *dual* system.

We did not perform model combination experiments for the raw input data, since we believed the pattern would be similar as for the clean data. The results for the raw devtest as input are considerably lower than for the clean data. Using the best model combination, we got a Bleu score of only 26.25, which can be compared to 29.90 using the clean data.

## 4 Data Cleaning Model

While the training data is noisy, we had access to cleaned versions of dev, devtest and test data. We decided to use the dev data to build a model for cleaning the noisy SMS data. We did this by training a standard PBSMT model from raw to clean dev data. When inspecting this translation model we found that it very often changed the place holders for names and phone numbers, and thus we filtered out all entries in the phrase table that did not have matching place holders. We then used this model to perform monotone decoding of the raw SMS data, thus creating a cleaner version of it.

This approach is similar to that of Aw et al.

| TMs | LMs | Bleu | NIST | Meteor |
|---|---|---|---|---|
| SMS | SMS | 29.04 | 5.578 | 52.32 |
| SMS | SMS, other | 28.76 | 5.543 | 51.96 |
| SMS | SMS, other+bible | 29.18 | 5.696 | 51.77 |
| SMS, other | SMS | 29.78 | 5.808 | 52.86 |
| **SMS, other+bible** | **SMS** | 29.90 | 5.764 | 52.88 |
| SMS, other+bible | SMS, other | 29.59 | 5.742 | 52.28 |
| SMS, other+bible | SMS, other+bible | 28.75 | 5.587 | 52.52 |

Table 2: Translation results, with different combinations of translation and language models. Model names separated by a comma stands for separate models, and names separated with a plus for one model built from concatenated corpora.

| Model | Testset | Bleu | NIST | Meteor |
|---|---|---|---|---|
| **Dual** | **clean** | 29.90 | 5.764 | 52.88 |
| Dual+CM | clean | 29.78 | 5.740 | 52.95 |
| Dual | raw | 26.25 | 5.231 | 50.79 |
| Dual | raw+CM | 26.26 | 5.348 | 51.30 |
| Dual+CM | raw | 25.64 | 5.120 | 50.01 |
| **Dual+CM** | **raw+CM** | 26.24 | 5.362 | 51.64 |

Table 3: Translation results, with and without an additional cleaning model (+CM) on the clean and raw devtest data

(2006), who trained a model for translation from English SMS language to standard written English, with very good results both on this task itself, and on a task of translating English SMS messages into Chinese. For training they used up to 5000 sentences, but the results stabilized already when using 3000 training sentences. Our task is different, though, since we do not aim at standard written Haitian, but into cleaned up SMS language, and our training corpus is a lot smaller, only 900 sentences.

Table 3 shows the results of using the cleaning model on training data and raw translation input. For the clean data using the cleaning model on the training data had very little effect on any of the metrics used. For the raw data translation results are improved as measured by NIST and Meteor when we use the filter on the devtest data, compared to using the raw devtest data. Using the filter on the training data gives worse results for non-filtered devtest data, but the overall best results are had by filtering both training and devtest data for raw translation input. Based on these experiments we used the cleaning model both on test and training data for raw input, but not at all for clean input, marked in bold in Table 3.

## 5 Spell Checking-based Replacement of Unknown Words

The SMS data is noisy, and there are often many spelling variations of the same word. One example is the word *airport*, which occur in the training corpus in at least six spelling variants: the correct *ayeropò*, and *aeoport, ayeopò, aeroport, aeyopòt,* and *aewopo*, and in the devtest in a seventh variant *ayéoport*. The non-standardized spelling means that many unknown words (out-of-vocabulary words, OOVs) have a known spelling variant in the training corpus. We thus decided to treat OOVs using a method inspired by spell-checking techniques, and applied an approximate string matching technique to OOVs in the translation input in order to change them into known spelling variants.

OOV replacement has been proposed by several researchers, replacing OOVs e.g. by morphological variants (Arora et al., 2008) or synonyms (Mirkin et al., 2009). Habash (2008) used several techniques for expanding OOVs in order to extend the phrase-table. Yang and Kirchhoff (2006) trained a morphologically based back-off model for OOVs. Bertoldi et al. (2010) created confusion networks as input of translation input with artificially created misspelled words, not specifically targetting OOVs, however. The work most similar to ours is DeNeefe et al. (2008), who also created lattices with spelling alternatives for OOVs, which did not improve translation results, however. Contrary to us, they only considered one edit per word, and did not weigh edits or lattice arcs.

Many standard spell checkers are based on the noisy channel model, which use an error (channel) model and a source model, which is normally mod-

eled by a language model. The error model normally use some type of approximate string matching, such as Levenshtein distance (Levenshtein, 1966), which measures the distance between two strings as the number of insertions, deletions, and substitutions of characters. It is often normalized based on the length of the strings (Yujian and Bo, 2007), and the distance calculation has also been improved by associating different costs to individual error operations. Church and Gale (1991) used a large training corpus to assign probabilities to each unique error operation, and also conditioned operations on one consecutive character. Brill and Moore (2000) introduced a model that worked on character sequences, not only on character level, and was conditioned on where in the word the sequences occurred. They trained weights on a corpus of misspelled words with corrections.

Treating OOVs in the SMS corpus as a spell checking problem differs from a standard spell checking scenario in that the goal is not necessarily to change an incorrectly spelled word into a correct word, but rather to change a word that is not in our corpus into a spelling variant that we have seen in the corpus, but which might not necessarily be correctly spelled. It is also the case that many of the OOVs are not wrong, but just happen to be unseen; for instance there are many place names. Thus we must make sure that our algorithm for finding spelling equivalents is bi-directional, so that it cannot only change incorrect spellings into correct spellings, but also go the other way, which could be needed in some cases. We also need to try not to suggest alternatives for words that does not have any plausible alternatives in the corpus, such as unknown place names.

### 5.1 Approximate String Matching Algorithm

The approximate string matching algorithm we suggest is essentially that of Brill and Moore (2000), a modified weighted Levenshtein distance, where we allow error operations on character sequences as well as on single characters. We based our weight estimations on the automatically created list of lexical variants that was built as a step in building the cleaning model, described in section 4. This list is very noisy, but does also contain some true spelling equivalents. We implemented two versions of the algorithm, first a simple version which used manually identified error operations, then a more complex variant where error operations and weights were found automatically.

#### Manually Assigned Weights

We went through the lexicon list manually to identify edits that could correct the misspellings that occurred in the list. We identified substitutions limited to three characters in length, and at the beginning and end of words we also identified letter insertions and deletions. The inspection showed that it was very common for letters to be replaced by the same letter but with a diacritic, or with a different diacritic, for instance to vary between [*e, é, è*]. Another common operation was between a single character and two consecutive occurrences of the same character. Table 4 shows the 46 identified operations. To account for the fact that we do not want our error model to have a directionality from wrong to correct, we allow operations in both directions.

Since the operations were found manually we did not have a reliable way to estimate weights, and used uniform weights for all operations. The operations in Table 4 have the weights given in the table, substitution of a letter with a diacritic variant 0, single to double letters 0.1, insertions and deletions 1 and substitutions other than those in the table, 1.6.

#### Automatically Assigned Weights

To automatically train weights from the very noisy list of lexical variants, we filtered it by applying the edit distance with the manual weights described above to phrase pair that did not differ in length by more than three characters. We used a cut-off threshold of 2.8 for words where both versions had at least six characters, and 1 for shorter words. This gave us a list of 587 plausible spelling variants, from the original list with 1635 word pairs.

To find good character substitutions and assign weights to them, we used standard PBSMT techniques as implemented in Moses, but on character level, with the filtered list of word pairs as training data. We inserted spaces between each character of the words, and also added beginning and end of word markers, e.g., the word *problém* was tokenized as '*B p r o b l é m E*'. Thus we could train a PBSMT system that aligned characters using GIZA++, and extracted and scored phrases, which in this case

| | Manual | | Automatic | |
|---|---|---|---|---|
| **Type** | **Instances** | **Weight** | **Examples+weights** | **Count** |
| mid 1-1 | e-i, a-o, i-y, a-e, i-u, s-c, r-w, c-k, j-g, s-z, n-m | .2 | n-m .90, e-c .74, j-g .62 | 12 |
| mid 1-2 | z-sz, i-iy, m-nm, n-nm, y-il, i-ye, s-rs, t-th, o-an, x-ks, x-kz, e-a, | .2 | x-ks .35, i-ue .83, w-rr .74 | 107 |
| mid 1-3 | – | – | e-ait .75 e-eur .66 | 29 |
| mid 2-2 | wa-oi, we-oi, en-un, xs-ks | .2 | we-oi .67, wo-ro .20, ie-ye .54 | 103 |
| mid 2-3 | wa-oir, ye-ier, an-ent, eo-eyo | .2 | iv-eve .79, ey-eyi .18 | 160 |
| mid 3-3 | syo-tio, syo-tyo | .2 | ant-ent .81, dyo,dia .67 | 116 |
| beg 0-1 | $\epsilon$-h, $\epsilon$-l | .2 | $\epsilon$-n .95, $\epsilon$-m .90, $\epsilon$-h .50 | 9 |
| beg 0-2 | – | – | $\epsilon$-te .95, $\epsilon$-pa .82 | 6 |
| beg 1-1 | h-l | .2 | a-e .89, w-r .67 i-u .33 | 5 |
| beg 1-2,3 | – | – | e-ai .68, a-za .74 k-pak .48 | 30 |
| beg 2,3-2,3 | – | – | wo-ro 0, ex-ekz .65, ens-ins .17 | 58 |
| end 0-1 | $\epsilon$-e, $\epsilon$-t, $\epsilon$-n, $\epsilon$-m, $\epsilon$-r, $\epsilon$-y | .2 | $\epsilon$-r .57 $\epsilon$-e .85, $\epsilon$-v .75 | 12 |
| end 0-2 | $\epsilon$-te, $\epsilon$-de, $\epsilon$-ue, $\epsilon$-le | 1 | $\epsilon$-de .93, $\epsilon$-le .75 | 7 |
| end 1-1 | – | – | e-o .74, n-m .86 | 5 |
| end 1-2,3 | – | – | i-li .81, c-se .62 n-nne .66 | 48 |
| end 2,3-2,3 | – | – | sm-me .67, ns-nce .38, wen-oin .36 | 70 |

Table 4: Error operations at the *mid*dle, *beg*inning and *end* of words. For manually defined operations all instances are shown, with their uniform score. For automaticcally identified operations examples are shown with their score, and the total count of each operation type.

amounts to creating a phrase-table with character sequences. The phrase probabilities are given in both translation directions, $P(S|T)$ and $P(T|S)$. Since we do not want our scores to have any direction, we used the arithmetic mean of these two probabilities to calculate the score for the pair, which is calculated as $1 - ((P(S|T) + P(T|S))/2)$, to also convert the probabilities to costs. To compensate for errors made in the extraction process, we filtered out phrase pairs where both probabilities were lower than 0.1.

To get fair scores for character sequences of different lengths we applied the phrase table construction four times, while increasing the limit of the maximum phrase length from one to four. From the first phrase table, with maximum length 1, we extracted 1-1 substitutions, from the second table 1-2 and 2-2 substitutions, and so on. We used the beginning and end of word markers both to extract substitutions that were only used at the beginning or end of sentences, and to extract deletions and insertions used at the beginning and end of words. Again, we only allowed substitutions up to three characters in length. The fourth phrase-table, with phrases of length four, were only used to allow us to extract substitutions of length three at the beginning and end of words, since the markers count as tokens. Table 4 shows the types of transformations extracted, some examples of each with their score, and the count of each transformation. A total of 777 operations were found, compared to only 46 manual operations. There were few substitutions with diacritic variants, so again we allowed them with a zero cost. The costs for deletions, additions, and substitutions not given any weights were the same as before, 1, 1, and 1.6. For the edit distance with the automatic weights, we used scores that were normalized by the length of the shortest string.

### Application to OOVs

We applied the edit distance operation on all OOVs longer than 3 characters, and calculated the distance to all words in the training corpora that did not differ in length with more than two characters. We used the standard dynamic programming implementation of our edit distance, but extended to check the scores not only in directly neighbouring cells, but in cells up to a distance of 3 away, to account for the maximum length of the character sequence substitutions. It would have been possible to use a fast trie imple-

| System | Clean devtest | | | Raw devtest | | |
|---|---|---|---|---|---|---|
| | **Bleu** | **NIST** | **Meteor** | **Bleu** | **NIST** | **Meteor** |
| No OOV treatment | 29.90 | 5.764 | 52.88 | 26.24 | 5.362 | 51.64 |
| Manual 1-best | 29.76 | 5.721 | 52.91 | 26.60 | 5.417 | 52.17 |
| Automatic 1-best | 29.90 | 5.746 | 52.83 | 26.26 | 5.351 | 51.60 |
| **Manual lattice** | 30.53 | 5.957 | 54.06 | 27.12 | 5.574 | 53.27 |
| Automatic lattice | 30.94 | 5.982 | 54.62 | 27.27 | 5.554 | 52.99 |
| Automatic lattice + LM | 30.33 | 5.912 | 54.07 | 27.79 | 5.555 | 52.98 |

Table 5: Translation results, using the approximate string matching algorithm for OOVs. The submitted system is marked with bold.

mentation (Brill and Moore, 2000), however.

We performed both 1-best substitution of OOVs, and lattice decoding where we kept the three best alternatives for each word. In both cases we only replaced OOVs if the edit distance scores were below a threshold of 1.2 for the manual weights, which were not normalized, and for the normalized automatic weights below 0.25, or below 0.33 for word pairs where both words had at least 6 characters. These thresholds were set by inspecting the results, but resulted in a different number of substitutions:

- clean (total 691)
  - manual: 251
  - automatic: 222
- raw (total 932)
  - manual: 601
  - automatic: 437

The lattice arcs were weighted with the edit distance score, normalized to fall between 0-1. We also tried to include a source language model score in the weights in the lattice, to account for the source model that has been shown to be useful for spelling correction, but which has not been found useful for OOV replacement. We trained a 3-gram language model on the Haitian SMS text, and applied this model for a five-word context around the replaced OOV. We used a single lattice weight where half the score came from the edit distance, and the other half represented the language model component. A better approach though, would probably have been to use two weights.

### 5.2 Results

Table 5 shows the results of the OOV treatment. When using 1-best substitutions there are small differences compared to the baseline on both test sets, except for the system with manual weights on raw data, which was improved on all metrics. All three ways of applying the lattice substitutions led to large improvements on all metrics on both test sets. On the clean test set it was better to use automatic than manual weights when not using the language model score, which made the results worse. On the raw test set the highest Meteor and NIST scores were had by using manual weights, whereas the highest Bleu score was had by using automatic weights with the language model. The system submitted to the workshop is the system with a lattice with manual weights, marked in bold in Table 5, since the automatic weights were not ready in time for the submission.

## 6 Conclusion

In this article we presented methods for translating noisy Haitian Creole SMS messages, which we believe are generally suitable for small and noisy corpora and under-resourced languages. We used an automatically trained cleaning model, trained on only 900 manually cleaned sentences, that led to improvements for noisy translation input. Our main contribution was to apply methods inspired by spell checking to suggest known spelling variants of unknown words, which we presented as a lattice to the decoder. Several versions of this method gave consistent improvements over the baseline system. There are still many questions left about which configuration that is best for weighting and pruning the lattice, however, which we intend to investigate in future work. In this work we only considered OOVs in the translation input, but it would also be interesting to address misspelled words in the training corpus.

## References


Karunesh Arora, Michael Paul, and Eiichiro Sumita. 2008. Translation of unknown words in phrase-based statistical machine translation for languages of rich morphology. In *Proceedings of the First International Workshop on Spoken Languages Technologies for Under-resourced languages (SLTU-2008)*, pages 70–75, Hanoi, Vietnam.

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, poster session*, pages 33–40, Sydney, Australia.

Nicola Bertoldi, Mauro Cettolo, and Marcello Federico. 2010. Statistical machine translation of texts with misspelled words. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the NAACL*, pages 412–419, Los Angeles, California, USA.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the ACL*, pages 286–293, Hong Kong.

Kenneth W. Church and William A. Gale. 1991. Probability scoring for spelling correction. *Statistics and Computing*, 1:93–103.

Steve DeNeefe, Ulf Hermjakob, and Kevin Knight. 2008. Overcoming vocabulary sparsity in MT using lattices. In *Proceedings of the 8th Conference of the Association for Machine Translation in the Americas*, pages 89–96, Waikiki, Hawaii, USA.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurence statistics. In *Proceedings of the Second International Conference on Human Language Technology*, pages 228–231, San Diego, California, USA.

Nizar Habash. 2008. Four techniques for online handling of out-of-vocabulary words in Arabic-English statistical machine translation. In *Proceedings of the 46th Annual Meeting of the ACL: Human Language Technologies, Short papers*, pages 57–60, Columbus, Ohio, USA.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Edinburgh, UK.

Philipp Koehn and Barry Haddow. 2009. Edinburgh's submission to all tracks of the WMT 2009 shared task with reordering and speed improvements to Moses. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 160–164, Athens, Greece.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic, June.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of the International Workshop on Spoken Language Translation*, Pittsburgh, Pennsylvania, USA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL, demonstration session*, pages 177–180, Prague, Czech Republic.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic.

Vladimir Iosifovich Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Shachar Mirkin, Lucia Specia, Nicola Cancedda, Ido Dagan, Marc Dymetman, and Idan Szpektor. 2009. Source-language entailment modeling for translating unknown terms. In *Proceedings 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*, pages 791–799, Suntec, Singapore.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 42nd Annual Meeting of the ACL*, pages 160–167, Sapporo, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 311–318, Philadelphia, Pennsylvania, USA.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the Seventh International Conference on Spoken Language Processing*, pages 901–904, Denver, Colorado, USA.

Mei Yang and Katrin Kirchhoff. 2006. Phrase-based backoff models for machine translation of highly inflected languages. In *Proceedings of the 11th Conference of the EACL*, pages 41–48, Trento Italy.

Li Yujian and Liu Bo. 2007. A normalized Levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095.