An Efficient Technique for De-noising Sentences using Monolingual Corpus and Synonym Dictionary

Sanjay Chatterji*, Diptesh Chatterjee, Sudeshna Sarkar
Department of Computer Sc. & Engineering, Indian Institute of Technology, Kharagpur, India
e-mail: {schatt, diptesh, sudeshna}@cse.iitkgp.ernet.in

Abstract
We describe a method of correcting noisy output of a machine translation system. Our idea is to consider different phrases of a given sentence, and find appropriate replacements of some of these from the frequently occurring similar phrases in the monolingual corpus. The frequent phrases in the monolingual corpus are indexed by a search engine. When looking for similar phrases we consider phrases containing words that are spelling variations of or are similar in meaning to the words in the input phrase. We use a framework where we can consider different ways of splitting a sentence into short phrases and combining them so as to get the best replacement sentence that tries to preserve the meaning meant to be conveyed by the original sentence.

1 Introduction

A sentence may contain a number of mistakes in the word level, phrase level and sentence level. These mistakes may be referred to as noise. Spelling mistakes, wrong lexical usage, use of inappropriate function words (like determiner, preposition, article, etc.), grammatical errors, wrong ordering of words in a sentence are some of the commonly encountered noises.

Noisy sentences are widely prevalent both in human generated sentences as well as sentences generated by a Natural Language Processing (NLP) system. The generation of noisy sentences by humans may be due to carelessness, lack of good language writing ability or lack of knowledge of spelling, vocabulary or grammar of the language. The systems which return natural language sentences as output, for example Machine Translation (MT) systems often make mistakes in the sentence. In this work, we have used a method to handle spelling errors, word choice errors, extra or missing word errors and reordering errors in the sentence using a monolingual corpus, a synonym dictionary, and a stemmer.

We have incorporated this method as a post-processing system for our Bengali to Hindi and Hindi to Bengali machine translation systems. Our base translation systems are imperfect and generate imperfect sentences. We analyzed the outputs of these systems and observed that a lot of noise can be corrected by applying this method. The evaluation results show that we are able to improve the performance of machine translation systems.

2 Related Work

Some work have also been done on the correction of errors of noisy sentence by finding the most appropriate replacement for each word or phrase. Willett and Angell (1983) have

*The author and the work are partially supported by Indian Language to Indian Language Machine Translation project sponsored by MCIT, DIT, Govt. of India.

Proceedings of COLING 2012: Demonstration Papers, pages 59–66,
COLING 2012, Mumbai, December 2012.

corrected the spellings of the words by finding the closest replacement candidate from the dictionary. The closeness of the dictionary word and the misspelled word is calculated using the count of the common trigram characters. Yannakoudakis and Fawthrop (1983) have divided the misspelled words into elements of spellings and replaced wrong elements by corrected elements.

Dahlmeier and Ng (2011) used Alternating Structure Optimization (ASO) technique for correcting grammatical errors in English article and preposition. Helping Our Own(HOO) shared task has been carried out in 2010, 2011 and 2012 to correct some particular classes of words like article, preposition, determiner, etc. of the English text and is reported by Dale and Kilgarriff (2010, 2011) and Dale et al. (2012).

The correction module has been used as pre-processing and post-processing stages of some machine translation systems. The rule based post-editing module proposed by Knight and Chander (1994) has used online resources for learning rules to correct the output of a Japanese-English machine translation system. Xia and Mccord (2004) has used automatically learned reordering rules in a hybrid English-French machine translation system.

3 Motivation

For correcting noisy sentences, similar to the approaches used in Statistical Machine Translation (SMT) systems, e.g., the IBM model (Brown et al., 1993; Koehn et al., 2003), may be used. But the development of a parallel corpus of noisy phrases and corresponding correct phrases is a time consuming task. However, instead of developing a parallel corpus, we wish to use monolingual corpus to improve the fluency of noisy sentences. For faithfulness, a synonym dictionary, a stemmer and phonetic mappings may be used in finding the phrases which preserves the actual meaning of the noisy phrases. The algorithm should have the ability to account for different classes of errors such as, Preposition, Postpossition or suffix errors, Spelling errors, Word form, Redundancy, Missing Word, Word Choice, Word ordering, etc.

4 Our Approach

Our approach to correcting noise in the sentences consists of correcting noise in the phrases of the sentence. For this, we split the sentence into small phrases. We make use of a n-gram language model obtained from a monolingual corpus. Frequent phrases in the language model that are similar to an input phrase are considered as candidates replacement for that phrase.

The function used to split the sentence into small phrases and for combining their candidates is discussed in Subsection 4.1 and the searching of the suitable candidate phrases in the corpus for the small phrases of the sentence is discussed in Subsection 4.2.

4.1 Spliting and Combining Phrases

Consider a sentence of N words: $S = w_1 w_2 \dots w_N$; where w_i is the i^{th} word of the sentence. A phrase in the sentence is of the form $P_{ij} = w_i w_{(i+1)} \dots w_j$, where $1 \leq i \leq j \leq n$. The length of P_{ij} phrase is $(j - i + 1)$. This phrase can be split into two phrases P_{il} and $P_{(l+1)j}$ in different ways for $i \leq l < j$. A m-word phrase can thus be split in 2 sub-phrases in $m - 1$ ways. Each of these sub-phrases may be further split in the same way.

While considering each phrase of the sentence if the phrase is a short phrase its replacement

candidates (candidates) can be found from the language model created from the monolingual corpus. For any phrase (short or long), we also consider combining the candidates of sub-phrases of every possible decompositions of that phrase. All these possible candidates will be considered for selecting the best candidate of the phrase. This module can be implemented using a dynamic programming method and a triangular matrix data structure. Each cell in the triangular matrix is a placeholder of a phrase of the sentence. An example triangular matrix for a four word sentence is shown in Figure 1.

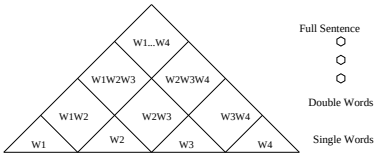

Figure 1: Triangular Matrix for a 4-word Sentence.

In the cell corresponding to a phrase, we store a list of candidates for that phrase. Candidate lists for lower length phrases of the sentence are stored at the lower level. In our bottom-up approach, members of the lower level cells are used to find the members of higher level cells. The basic algorithm is presented in Algorithm 1.

Algorithm 1 DynamicFind(Sentence) // Finds corrected forms of the Sentence.

INPUT: Sentence $S = w_1, w_2, ..., w_N$
Data Structure: R = Upper Triangle of a Square Matrix, $K = 5$
for n=1 to N **do** {/*n indicates the length of the phrase*/}
  for start=1 to $N - n + 1$ **do** {/*start is the starting index of the n length phrase*/}
    if $n == 1$ **then** {/* Phrase of length 1 */}
      R[start][start] = FIND_BEST_SUB($(P_{start,start}, K)$; /* Returns K candidates of $P_{start,start}$ phrase */
    else
      end = start+$n - 1$; /* end is the ending index of the n length phrase */
      if $n \leq k$ **then** {/*Short phrase*/}
        R[start][end] ← FIND_BEST_SUB($P_{start,end}, K$)
      end if
      for j=start to end-1 **do**
        P1 ← R[start][j]; P2 ← R[j+1][end]; P3 ← COMBINE(P1,P2)
        R[start][end] ← KBEST(P3, R[start][end],K)
      end for
    end if
  end for
end for
Return R[1][N]

COMBINE(P1,P2): Concatenates all the phrases of $P1$ with all the phrases of $P2$ and combine their corresponding values.
KBEST(L_1,L_2,K): Finds K best members among the members of two lists: L_1 and L_2.

In this algorithm, S is an input noisy sentence of N words. $P_{start,end}$ is a phrase, where

start and *end* are starting and ending indices in the sentence. n is the length of the phrase, where $n = end - start + 1$. For each short phrases the candidates along with their values computed by the substitute method discussed in the next subsection are stored in the corresponding bottom level cells of R.

Each multi word phrase is broken into a pair of sub-phrases. The variable j indicates the intermediate point or partition point of the phrase. So, two sub-phrases of $P_{start,end}$ are: one of index *start* to j and another of index $j + 1$ to *end*. A pair of candidate lists for each pair of sub-phrases are taken from previous level cells of the triangular matrix and stored in $P1$ and $P2$, respectively. The `COMBINE` function concatenates all the members of $P1$ with all the members of $P2$ and combine their values and store in $P3$.

For each short multi-word phrases, two candidate lists computed by substitute method and `COMBINE` function are sorted and top K are selected by the `KBEST` function. These are stored into the corresponding cells of the R matrix. The algorithm returns the members of the top cell of the R matrix as the most relevant corrected sentence.

4.2 Computing the Phrase Substitutes

All phrases of length 1 to K of monolingual corpus are stored in a language model along with their frequencies. Given a short phrase from the noisy sentence, we have to search for the short phrases in the language model which are frequent and similar to the noisy sentence phrase. These searched phrases are candidates of the noisy sentence phrase.

4.2.1 Scoring Function

For each short phrase of the sentence, we define the following scoring function, based on the practical scoring function defined by Hatcher et al. (2010) in Lucene search engine, to find suitable candidate phrases from the language model.

$$score(q, p) = coord(q, p) \sum_{t \in q} \{tf(t, p)\ idf(t)^2\}\ BoostValue(p, l) \quad (1)$$

Here, q is the query phrase of the sentence. p is a phrase of length l in language model, which is being considered currently. The components of equation 1 are explained below.

(i) Coordination factor $coord(q, p)$ indicates how many of query terms (words) are found in p. Length of query phrases and values of this function are both between 1 to k.

(ii) For term frequency $tf(t, p)$ we use square root of frequency of the query term t in p.

(iii) For inverse document frequency $idf(t)$ we have used the inverse of the number of phrases in which the query term t appears.

(iv) As we have mentioned earlier, we want to find those similar phrases which have highest frequency in the language model. So, we want to boost the score according to the frequency of the phrase. However, frequencies of the shorter phrases are not directly comparable with that of the longer phrases. Therefore, boost of a phrase is calculated with the help of the frequency of that phrase in the corpus and the length of the phrase.

4.2.2 Handling Variations

For finding suitable candidate phrases for each short phrase of noisy sentence, it does not suffice to only search for the frequent exact phrases in the language model which are similar to the input phrase. We need to search other variations of the words of this short phrase, e.g., spelling variation, morphological variation and lexical variation. We have developed a unified approach to handle all these effectively. This approach consisting of indexing several variations of each input phrase, and considering these variations for retrieval.

Indexing
This is done by indexing each phrase at several different levels, apart from indexing the original phrase. At the phonetic level, a phrase consisting of phonetically normalized words of the original phrase is indexed. At the morphological level, the phrase consisting of stemmed words of the original phrase is indexed, and at the lexical level, the phrase consisting of the synonym group IDs of the words in the original phrase is indexed.

Retrieval
Given an input phrase, K similar phrases are retrieved at various levels – unaltered, morphological, phonetic, and lexical. We define four coefficients: c_{word}, c_{stem}, c_{pm} and c_{id} for these four searches, respectively and multiplied the scores of the searches with the corresponding coefficients. The top K phrases at each level are identified by using the scoring function shown in equation 1.

5 Experimental Results

In our experiments, we have considered that the length of short phrases is between 1 and 5. Further, the length of the indexed phrases is between 1 and 5. The values of c_{word}, c_{stem}, c_{pm} and c_{id} coefficients are experimentally set as 10, 8, 6, and 6, respectively. The value of K is considered as 10.

5.1 Experimental Setup

The proposed approach is implemented for Hindi and Bengali languages. We have used following resources and modules for this task.

(i) Hindi and Bengali monolingual corpus crawled from the web and cleaned. The size of these two corpus are 300K and 430K sentences, respectively.

(ii) Hindi and Bengali synonym dictionaries divide words into some groups of synonyms.

(iii) Hindi and Bengali longest suffix stripper for stemming.

(iv) Hindi and Bengali words to phonetic word map table similar to the one proposed by UzZaman and Khan (2005).

5.2 Analysis with example

We now show how the system performed on some Hindi written sentences. The Hindi sentences are represented in itrans format with the word by word translation in square bracket. For every Hindi sentence, we show the English translation of the intended sentence. Then, the final sentence output by our system is given where the correct modifications are

shown in boldface, and the incorrect modifications are underlined. Finally, we present a summary of the modifications.

1. **Original Hindi sentence (OH) :** mujhe le.DakA **ko** pasanda AYA. [To-Me Boy To Like Came]
   Intended meaning in English (E) : I liked the boy.
   Final sentence (FH) : mujhe le.DakA pasanda **AyA**. [To-Me Boy Like Came]
   Summary of Modifications(SM) : Deleted postposition 'ko'. Changed spelling of 'AYA'.

2. **OH:** merA hindI AchchhA nahI. [My Hindi Good Not]
   E: My Hindi is not good.
   FH: merA ye hindI **achchhA** nahI **hai**. [My This Hindi Good Not is]
   SM: Inserted pronoun 'ye' and verb 'hai'. Changed spelling of 'AchchhA'.

3. **OH:** mere yahA.N Ane kA kAraNa hai mai.N khAnA chAhatA hai. [My Here Come 's Reason Is I Eat Want Is]
   E: The reason why I came here is I wanted to eat something.
   FH: mere yahA.N Ane kA kAraNa **yaha** hai **ki** jo mai.N khAnA chAhatA **thA**. [My Here Come 's Reason This Is That What I Eat Want Was]
   SM: Inserted 'yaha', 'ki' and 'jo'. Replace 'hai' by 'thA'.

5.3 An Application to postprocessing of sentences obtained through Machine Translation

The proposed approach is applied to correct the output of two imperfect transfer based Machine Translation systems, the BHMT system translates Bengali sentences to Hindi, whereas the HBMT system translates from Hindi to Bengali. The proposed approach has improved the quality of the output of both the systems in terms of both perplexity and BLEU scores. The results can be seen in Table 1.

	BHMT system	Modified System	HBMT system	Modified System
Perplexity	34.243	27.811	39.612	35.862
BLEU	0.232	0.243	0.218	0.227

Table 1: Scores of MT output and proposed modifications.

6 Conclusion

We have presented a unified approach for correcting different types of noise in a sentence. We are able to handle major classes of noise, though we are not able to handle long range re-orderings. This approach is general and can be used for any language. The resources needed are minimal and can be easily obtained.

There is a need for more experiments, better tuning of the scoring model, and testing on different sources of noisy sentences.

Acknowledgments
We would like to thank all the annotators of Communication Empowerment Lab, IIT Kharagpur, for their active participation in the development of corpus. We extend special thanks to Dr. Rajendra Prasath for helping in data mining related tasks.

References

Brown, P. F., Pietra, V. J., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Dahlmeier, D. and Ng, H. T. (2011). Grammatical error correction with alternating structure optimization. In *ACL*, pages 915–923.

Dale, R., Anisimoff, I., and Narroway, G. (2012). Hoo 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62, Montréal, Canada. Association for Computational Linguistics.

Dale, R. and Kilgarriff, A. (2010). Helping Our Own: Text massaging for computational linguistics as a new shared task. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 261–266, Dublin, Ireland.

Dale, R. and Kilgarriff, A. (2011). Helping Our Own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 242–249, Nancy, France.

Hatcher, E., Gospodnetic, O., and McCandless, M. (2010). *Lucene in Action*. Manning, 2nd revised edition.

Knight, K. and Chander, I. (1994). Automated postediting of documents. In *AAAI*, pages 779–784.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

UzZaman, N. and Khan, M. (2005). A double metaphone encoding for bangla and its application in spelling checker. In *Proc. 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering*.

Willett, P. and Angell, R. (1983). Automatic spelling correction using a trigram similarity measure. *Information Processing & Management 19*, pages 255–261.

Xia, F. and Mccord, M. (2004). Improving a Statistical MT System with Automatically Learned Rewrite Patterns. In *Proceedings of Coling 2004*, pages 508–514, Geneva, Switzerland. COLING.

Yannakoudakis, E. J. and Fawthrop, D. (1983). The rules of spelling errors. *Information Processing And Management*, 19(2):87–99.