

Word Sense Disambiguation for Improving Accuracy of Machine Translation using Translation Memory

¹Sunita Rawat, ²Manoj Chandak, ³Tabassum Khan

¹Research Scholar, G. H. Rasoni College of Engineering, Nagpur
Email: ssunitarawatt@gmail.com

²Supervisor, G. H. Rasoni College of Engineering, Nagpur
Email: chandakmb@gmail.com

³Assitant Professor, G. H. Rasoni College of Engineering, Nagpur
Email: tabassum.khan@raisoni.net

Abstract

Problem of Word Sense Disambiguation (WSD) is about selecting the correct sense of an ambiguous word in a given context. N-grams are successive coinciding N-character string structured from an input sentence. In wordsense disambiguation these models are widely applied. In this paper Machine Translation System (MTS) that uses the N-gram model is considered. To minimize the responsetime of machine translation system, we plan to use a translationmemory (TM) for machine translation. The incorporated framework of machine translation system with translation memory defined. To additionally accelerate the translation process, machine translation system is combined with the Hadoopframework and performed distributed operation.

Keywords: Word Sense Disambiguation, POS tagging, translation memory, Machine Translation System, N-gram model, Hadoop framework.

1. Introduction

Word Sense Disambiguation is very common and complicated difficulty in Natural Language Processing (NLP). Nearly in all natural languages several words have various meanings. Consider the following sentence written in English language-

- (a) Most of the people in India put fast on the occasion of Mahashivratri.
- (b) Saumya runs very fast.

In the first sentence the word 'fast' is related to not have food while in the second it is related to speed. Techniques of word sense disambiguation are broadly used in Machine Translation (MT) to find the proper meaning of the word in the given scenario.

POS tagging or part-of-speech tagging is the procedure of assigning a grammatical category like noun, verb, adjective etc. to a word and becomes key factor when a word is polysemous. In computer linguistics, part-of-speech (POS) tagging is a process of marking a word in a script or corpus using its definition and keeping in mind its context used in the sentence and its relationship with adjacent words, phrases and related words in the corpus. It can also be called grammatical tagging. Identifying the nouns, verbs, adjectives, adverbs and tagging them to the words.

The translation memory is a database consists of so many translations, which are stored in the form of "segment pairs" i.e. source text is stored collectively with the equivalent target text. As per earlier defined rules segments are formed by dividing the text sections. Segment ends are indicated by the markers. Segments are thus generally table elements, list or sentences. Whichever new text comes for translation is first get compared with the previously available segments in the translation database. The TMS decides similarity of segments (source language) and recommends the stored, corresponding segment (existing translation) in the target language. Analogous segments are also identified and suggested as alleged fuzzy matches.

2. Related Works

The selection of the appropriate sense for a given ambiguous word is commonly carried out by considering the words surrounding the ambiguous word. Initially Bar-Hillel (1964) tried word sense disambiguation for Machine Translation [6]. According to him "Universal Encyclopedia" is

must for a machine to enable it to differentiate among the several meanings of a word. Brown and Pietra (1992) used a very bigger text of English language and by using trigram ($n=3$) model set a higher limit of 1.75 bits per character [7]. Brown, Della Pietra S.A, Della Pietra V.J., and Mercer (1993) created five statistical models for translating a complete sentence from one language to another and found that it is feasible to guess their parameter automatically from a stored set of translated pairs [8]. They have also experimented and found that algorithmically it is feasible to line up the words inside pairs of sentences. Primarily they have worked on English-French MTS [5]. Although their methods had least linguistic matter so these can perform correct on rest of the pairs of languages.

In the late 1960s and early 1970s the fundamental ideas about translation memory came up, with the objective to offer translators with various resources that would permit them to utilize it as a reference tool to perform their task more precisely and efficiently. In 1980s due to growth in the processing power and storage capacity of personal computers numerous commercial computer-aided translation tools have certainly developed significantly [1,2]. For pattern recognition and segmentation algorithm these tools can be used. Due to existence of enormous multilingual corpora, extensive variety of domains enclosed and require fast response time have responsible for the technology to go in the direction of combination of translation memory with machine translation system. Lots of work has been accounted till date where Example-Based Machine Translation (EBMT) systems and Statistical Models [3] use translation memory for fast translation.

Zhechev and Genabith have published a further work on SMT [4]. It is phrase-based sub-tree arrangement to determine parts of the source sentence that match up to parts of the recommended translation took out from the database of translation memory. This method develops alignment algorithm for Fuzzy-Match Score and phrases for greatest match selection.

3. Proposed work

System mainly consists of three modules. First module is word sense disambiguation. Second module contains translation memory and last module includes Hadoop framework. We approached the word sense disambiguation task by means of a probabilistic system based on Naïve Bayes, which considers the probability of a word sense (in the target language), given a sentence (in the source language) containing the ambiguous word. We calculate the probability of each word in the source language of being associated/ translated to the corresponding word (in the target language).

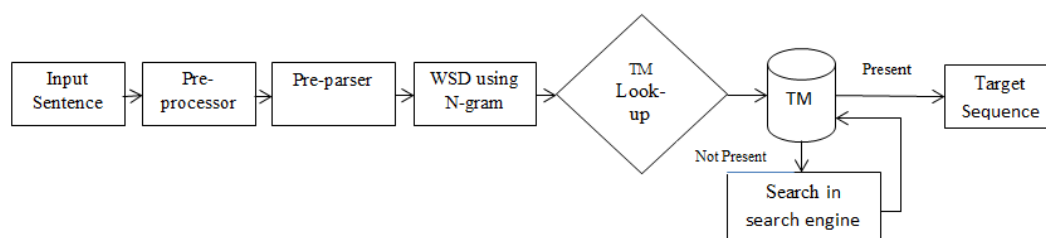


Figure 1: Architecture of the proposed system

While in translation process translation memory is used to look up each input segment in that translation memory. If exact match is available, system will give translated part of that matched segment as an advice for reuse to the translator. In case exact match is not available, system will try to find somewhat similar segments. As the input string not matches fully but some percentage, so it is known as fuzzy matches. After this translator takes the decision, whether to reuse the already translated text or modify it to form new translation and save it to database. Therefore translator, use to make segment pairs that may be available for future reference [10,12].

For large scale data processing we have used Hadoop framework. It is a broadly used open source framework having basis of distributed batch processing. We have considered 20,000 sentences, translation of such a huge input file is a monotonous job for the offline machine translation system. To complete this translation it requires about 20 days on a desktop machine. To utilize the resources provided by Hadoop framework, the translation system is one of the best applications [9]. Therefore,

we have used translation memory to accelerate speed of online translation, and Hadoop framework to accelerate translation speed for offline process.

4. Implementation

Input statements will go through the steps like tokenization, part of speech tagging, lemmatization, chunking and parsing. Following example shows the tagging and parsing of the given statement.

a) POS tagging of English sentence from agriculture domain.

1. Input Sentence : farmer is working in the rice farm
2. Pre-Processed output : 0=farmer 1=is 2=working 3=in 4=rice farm
3. Pre-Parsed source structure: 0=farmer-NN 1=is-VBZ 2=working-VBG 3=inrice farm-NN
4. Parsing and generation process
5. Generated Target Sequence: 0=NN-किसान 3=NN-चावलखेतमें 2=NP-काम 1=VGF-कर रहा है

Tagging

farmer/NN is/VBZ working/VBG in/IN rice/NN farm/NN

Parse

```
(ROOT
  (S
    (NP (NN farmer))
    (VP (VBZ is)
      (VP (VBG working)
        (PP (IN in)
          (NP (NN rice) (NN farm)))))))
```

b) POS tagging of Hindi sentence from agriculture domain.

किसान चावल खेत में काम कर रहा है

```
<Sentence id="1">
1  ((      NP  <fs af=किसान,n,m,sg,3,d,0,0' head="किसान">
1.1  किसान  NN  <fs af=किसान,n,m,sg,3,d,0,0' name="किसान">
      ))
2  ((      NP  <fs af=चावल,n,m,sg,3,d,0,0' head="चावल">
2.1  चावल   NN  <fs af=चावल,n,m,sg,3,d,0,0' name="चावल">
      ))
3  ((      NP  <fs af=खेत,n,m,sg,3,d,0,0' vpos="vib1_2" head="खेत">
3.1  खेत     NN  <fs af=खेत,n,m,sg,3,d,0,0' name="खेत">
      ))
4  ((      NP  <fs af=काम,n,m,sg,3,d,0,0' head="काम">
4.1  काम     NN  <fs af=काम,n,m,sg,3,d,0,0' name="काम">
      ))
5  ((      VGF <fs af=कर,v,m,sg,2,,0_रह+या_है,0' vpos="tam1_2_3" head="कर">
5.1  कर      VM  <fs af=कर,v,any,any,any,,0,0' name="कर">
      ))
</Sentence>
```

c) N-gram Model:

N-gram algorithm uses probabilistic methods. It applied for word prediction to predict next word after examining n-1 previous words. It is basically a series of n words together with their count i.e. number of times those words appear in training dataset.

In our system we have considered only bigram and trigram language model. Bigram, as well known as Markov assumption, believes that we can guess the probability of the next coming word just by looking at the last word came across. We can derive bigram to trigram by looking previous two words in the past. This can again be get generalize to N-gram by checking N-1 words in the past. Therefore, the common equation for finding the probability of the next coming word in a series is:

$$P(W) = \prod_{i=1}^{|W|+1} P(w_i | w_0 \dots w_{i-1})$$

Consider an example to calculate the probability by bigram model of word b following a. Count the bigrams c(ab) from the available corpus and normalize it with the total number of bigrams that begins with a.

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$C(w_{i-1})$ in the denominator shows the total count of bigrams starting with w_{i-1} as the bigrams beginning with w_{i-1} is same in number as that of w_{i-1} available in considered corpora. For example,

<s> I am farmer </s>

<s> farmer I am </s>

<s> I do not like fast food </s>

- 1) $P(I | <s>) = 2/3 = .67$
- 2) $P(\text{farmer} | <s>) = 1/3 = .33$
- 3) $P(\text{am} | I) = 2/3 = .67$
- 4) $P(<s> | \text{farmer}) = 1/2 = 0.5$
- 5) $P(\text{farmer} | \text{am}) = 1/2 = .5$
- 6) $P(\text{do} | I) = 1/3 = .33$

In the above first example probability of “starting sentence with I” is .67. As among considered 3 sentences 2 are starting with I. In third example probability of “I followed by am” is .67. As in above example in 3 sentences “I” is there but only in 2 sentences it followed by “am”.

Table 1 Probability Matrix:

	i	want	to	sow	wheat	seeds
i	5	827	0	9	0	0
want	2	0	608	1	6	86
to	2	0	4	686	0	0
sow	0	0	1	0	42	175
wheat	15	0	49	0	0	36
seeds	12	0	76	0	0	0

As shown in above matrix, possibility of occurrence of “I” with “want” is too much whereas “I” never comes with “to”. In same way “want” and “want” never come together whereas chances of finding “want” and “to” together is very high.

d) Translation Memory:

In translation memory we store parsed and generated output, these can be reuse repetitively based on lexical items present by carrying a TM lookup therefore no need of re-parsing and re-generation. During translation process TM look up for every segment available in that translation memory for perfect match. If an accurate match is available system will give translated text of input string and recommend the translator to reuse it. If only some part of text matches this comes under fuzzy match. In this case translator has to take decision either to use the existing translated text with modifications and store it back to translation memory or start translation from scratch and store it to translation memory database so that it can be use in future. Translation memory increases the efficiency of the system by saving processing time for translation as it uses already translated text if it matches to particular extent. Hence, system speed will increase as the size of translation memory grows.

e) Hadoop Framework:

It provides distributed storage and computation across clusters of computers. It is designed to scale up from single server to thousands of machines, each offering local computation and storage. It improves the speed of machine translation. Master node divides the sentences into chunks and distributes it among slave nodes. In Hadoop complete system works in parallel. Every slave node has its independent configured databases in order that every TaskTracker can use it and stay away from the congestion occurred in case of a central database. Aggregation of all local databases is done on Master node to form a translation memory for online machine translation system. Therefore, translation can be speed up with the larger size of translation memory which enhances machine translation response time, throughput and efficiency of the system.

5. Advantages

- **Accelerated Translation Speed**

Translation memory accelerates translation speed by reusing the text which has already been translated. So if text is already translated once then no need to start translation from scratch, it reduces time required for translation.

- **Reduced Translation Costs**

When a match is discovered, per-word translation rates are reduced in proportion to the strength of the match.

6. Conclusion

Word Sense Disambiguation is a very complicated job of artificial intelligence as it deals with the difficulties of NLP. It targets to recognize a semantic formation from unstructured resources of text. Basic reason for its intrinsic complexity is that it is extremely dependent on information gained from numerous sources. Therefore it is really hard for a computational system to process. In today's time there is a huge pressure on industries for translating text. To reduce this pressure translation memory is combined with machine translation system for translating text more precisely and efficiently. In this paper, along with concept of translation memory we have also used Hadoop framework. So, both online and offline translation will get perform very efficiently.

References

- [1] The LOGOS Intelligent Translation system. Available at <http://www.mtarchive.info/AMTA-1994-bennett.pdf>.
- [2] Wang K., Zong C., and Su K., 2013. Integrating Translation Memory into Phrase-Based Machine Translation during Decoding. In The 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013), Sofia, Bulgaria, August 4-9, 2013.
- [3] Ergun B. and Dymetman M., 2008. Dynamic TM: Using Statistical Machine Translation to Improve TM. In Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing), volume 4919 of Lecture Notes in Computer Science, pp. 454–465, 2008.
- [4] Zhechev V. and Genabith J.V., 2010. Seeding Statistical Machine Translation with TM Output through Tree-Based Structural Alignment. In SSST-4 4th Workshop on Syntax and Structure in Statistical Translation, pp. 43–51, 28 August 2010, Beijing, China.
- [5] Rakesh Kumar, Vishal Goyal and Ravinder Khanna, 2013. N-gram Based Word Sense Disambiguation of Hindi Post Position से (sē) in the context of Hindi to Punjabi Machine Translation System. In An International Journal of Engineering Sciences, pp. 59-67, Issue December 2013, Vol.9 ISSN: 2229-6913 (Print), ISSN: 2320-0332 (Online).

- [6] Bar-Hillel, Y. (1964). On Syntactic categories, *Journal of Symbolic Logic* 15.1-16 [Repr. Bar-Hillel 1964a, 19-37].
- [7] Brown, P.F., Pietra and S.A.D. (1992). An Estimation of Upper Bound for the Entropy of English, *Association for Computational Linguistics*, Volume 18, Number 1, pp. 31-40.
- [8] Brown, P.F., Della Pietra, S.A. Della Pietra, V.J., and Mercer R.L. (1993). The Mathematics of Statistical Machine Translation: Parameter estimation. *Computational linguistics* 19(2), 263–311.
- [9] Hadoop Framework. Available at <http://Hadoop.apache.org/>
- [10] Wang K., Zong C., and Su K., 2013. Integrating Translation Memory into Phrase-Based Machine Translation during Decoding. In: *The 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, Sofia, Bulgaria, August 4-9, 2013.
- [11] Jurafsky, D., Martin, J.H., 2013. *Speech and Language Processing*. Pearson, Eighteenth Impression.
- [12] V N Shukla, Nishtha Jaiswal and Seema Shukla, 2015. An Approach to Integrate Translation Memory in MT System for English to Indian Language, In *Proceedings of An International Conference on Cognitive Computing and Information Processing (CCIP)*, 2015.

