Disambiguating Hindi Words Using N-Gram Smoothing Models

UmrinderPal Singh[1], Vishal Goyal[2], Anisha Rani[3]

[1,2,3]Department of Computer Science, Punjabi University, Patiala, Punjab

[1,2,3] umrinderpal@gmail.com,vishal.pup@gmail.com, garganisha17@gmail.com

***Abstract*—**Word sense disambiguation is widely studied and discussed area of NLP for any natural language under consideration. Words have different senses. The task of selecting the correct sense for a word is called word sense disambiguation. In Machine Translation the problem when we translate the ambiguous words. To resolve this problem we develop Word Sense Disambiguation module that resolve the problem of ambiguity of Hindi words of a particular sentence. Deleted Interpolation and Back-Off, N-gram smoothing models are used to implement Word Sense Disambiguation. we have used the tri-gram to implement these models. we use a dictionary based approach.

Index Terms*—Word Sense Disambiguation, Punjabi, Back-Off, Deleted Interpolation*

I. INTRODUCTION

Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction. Many challenges in NLP involve natural language understanding that is, enabling computers to derive meaning from human or natural language input.[L1]

Word Sense Disambiguation is the process of identifying the correct sense of a word in a particular sentence, when the word has multiple meanings. For Example "वीर"(veer) *means* "बहादुर"(brave) or "भाई"(brother). WSD module should be used in order to obtain the correct sense of the ambiguous word in the sentence. In the fields of computational linguistics and probability, an *n*-gram is a contiguous sequence of *n* items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The *n*-grams typically are collected from a text or speech corpus[L2].

An N-gram model uses the previous n-1 words to predict the next one. N-gram model can be trained by counting and normalizing. Normalizing means dividing by some total count so resulting probabilities fall legally between 0 and 1. An n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram"; size 3 is a "trigram". N-Grams with N > 3 are not practical, because the number of parameters of an N-gram is V N, where V is the size of the vocabulary[1].

II. PREVIOUS WORK

Much of the early work in WSD was carried out within the context of MT in the 1950's. The earliest approach was by Weaver (1949), where he argued the need for WSD in MT, as described in his memorandum. Lehal et.al. [2] described Size of N for Word Sense Disambiguation using N gram model for Punjabi Language **.**In this research paper, tried to find out whether higher order n gram models improves the word sense disambiguation in Punjabi language and whether it has any relation with entropy of the models . Instead of generating a higher order model of n gram, which is time consuming, hard to create, we can make use of combination of lower order n gram model.Marti et. al.[4] tested different vocabulary size and concluded that language models become more powerful in recognition tasks with larger vocabulary size. Manish Sinha et.al. [5] Explained in their research paper "Hindi word sense disambiguation" that WSD for Hindi words make use of the Wordnet for Hindi developed at IIT Bombay, which is a highly important lexical knowledge base for Hindi. This is the first attempt for an Indian language at automatic WSD. The accuracy value ranges from about 40% to about 70%. The performance can surely be improved if morphology is handled exhaustively. This system only deals with nouns.

III. N-GRAM SMOOTHING MODELS

An N-gram is simply a sequence of successive n words along with their count i.e. number of occurrences in training data. For computational reasons, Markov assumptions are applied which states that current word does not depends on the entire history of the word but at most on the last few

words . The number of words in the local context of ambiguous words makes a window. The size of window i.e. number of words to be considered at ± n positions is important. Instead of generating a higher order model of n gram, which is time consuming, hard to create and maintain, and of course need a lot of data to get meaningful results, we can make use of combination of lower order n gram model[2].

This task of reevaluating some of the zero-probability and low-probability *N*-grams, and assigning them non-zero values, is called smoothing. Various methods are used for smoothing the probability distribution. We are describing some of them.

A. Add-one Smoothing

Add-One Smoothing is also called Laplace Smoothing. This is a very simple technique where we add one in all the counts. Add-One smoothing is defined as following:

Bigram: $P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$

N-Gram: $P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)+1}{C(w_{n-N+1}^{n-1})+V}$

Where V is the total number of possible (N-1) grams (i.e. the vocabulary size for a bigram model). In Add-One Smoothing method too much mass is shifted towards unseen data and all unseen n-grams are smoothed in the same way.

B. Good -Turing Smoothing

Basic Idea of Good-Turing Smoothing is use the count of things you've seen once to help estimate the count of things you've never seen. Specifically, judge rate of 0-count events by rate of 1-count events. Estimate probability of things which occurs c times with the probability of things which occur c+1 times defined as:

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

$$P_{GT}^*(words\ with\ 0\ Freq) = \frac{N_1}{N}$$

C. BACKOFF

When you are using N-Gram model you may come up with 0.0 probabilities for those words which are not part of the training data like when you encounter with Proper names and new words. When we are estimating the Maximum Likelihood from training data.

$$P_{ML}(x) = \frac{C(x)}{\sum_e c(e)}$$

Because of insufficient training data there are events x that may be zero $C(x) = 0$, so that ML will be $P_{ML}(x) = 0$, and our language model assign probability zero to unseen words. While calculating the probabilities using Maximizing Likelihood Estimation (MLE), we need to smooth the probability distribution and assign non-zero values to unseen words.

Backoff *N*-gram modeling is a nonlinear method introduced by Katz (1987). In the backoff model, we build an *N*-gram model based on an *(N-1)*-gram model. The difference is that in backoff, if we have non-zero trigram counts, we rely solely on the trigram counts and don't interpolate the bigram and unigram counts at all. We only 'back off' to a lower-order *N*-gram if we have zero evidence for a higher-order *N*-gram. If we have no examples of a particular trigram

$$W_{n-2}W_{n-1}W_n$$

to help us compute trigram probability.

$$p(W_n|W_{n-2}W_{n-1})$$

$$p(W_n|W_{n-2}W_{n-1}) = \frac{Count(W_{n-2}W_{n-1}W_n)}{Count(W_{n-2}W_{n-1})}$$

Similarly, if we don't have counts to compute, we can look to the Bigram Model.

$$p(W_n|W_{n-1}) = \frac{Count(W_{n-1}W_n)}{Count(W_{n-1})}$$

Back-off to Unigram if Bigram not exists.

$$p(W_i) = \frac{Count(W_i)}{\sum_{i=0}^{n} Wi}$$

Specifically, for the case of trigram:

$$p(W_n|W_{n-2}W_{n-1}) = \begin{cases} p(W_n|W_{n-2}W_{n-1}), if\ count(Count(W_{n-2}W_{n-1}W_n) > 0 \\ p(W_n|W_{n-1}), if\ Count(W_{n-1}W_n) > 0 \\ Count(W_i), otherwise \end{cases}$$

D. DELETED INTERPOLATION

The deleted interpolation algorithm, due to Jelinek and Mercer (1980), combines different *N*-gram orders by linearly interpolating all three models whenever we are computing any trigram. That is, we estimate the probability by mixing together the unigram, bigram, and trigram probabilities. Each of these is weighted by a linear weight [3].

$$p(w_i|w_{i-2}w_{w-1}) = \lambda_1 p(w_i|w_{i-2}w_{w-1}) + \lambda_2 p(w_i|w_{w-1}) + \lambda_3 p(w_i)$$

where $\sum \lambda_1 + \lambda_2 + \lambda_3 = 1$

Training the λ's can be done so as to maximize the likelihood of a held-out data, separate from the main training corpus. we use unigram $\lambda_5 p(w_i)$ and left-bigram $\lambda_3 l\_p(w_i|w_{w-1})$, right-bigram $\lambda_4 r\_p(w_i|w_{w-1})$, left-trigram $\lambda_1 l\_p(w_i|w_{i-2}w_{w-1})$ and right-trigram $\lambda_2 r\_p(w_i|w_{i-2}w_{w-1})$ probabilities to estimate probability of trigram. Left-bigram is that see one word previous the ambiguous word and right-bigram is that see one word after ambiguous word. Similarly left-trigram sees two words previous the ambiguous word and right-trigram see two words after the ambiguous word.

$$p(w_i|w_{i-2}w_{w-1}) = \lambda_1 l\_p(w_i|w_{i-2}w_{w-1}) + \lambda_2 r\_p(w_i|w_{i-2}w_{w-1}) + \lambda_3 l\_p(w_i|w_{w-1}) + \lambda_4 r\_p(w_i|w_{w-1}) + \lambda_5 p(w_i)$$

$$\lambda_1 = \lambda_2 = 2.5, \qquad \lambda_3 = \lambda_4 = 1.77, \qquad \lambda_5 = 1.1$$

IV. RESULTS AND DISCUSSION

The accuracy of any Word Sense Disambiguator is measured by Word Sense Disambiguation rate. Word Sense Disambiguation rate is defined as percentage of words which are correctly disambiguated in the Translation Language model.

$$WSD\ rate = \frac{Total\ No.of\ Words\ Translated\ Correctly}{Total\ no.of\ Words}$$

We have used database contain Hindi words along with meaning of ambiguous word in Punjabi. The database contain 10 highly polysemous Hindi words.

Accuracy of Word Sense Dis-ambiguator with different N-gram Smoothing models.

Technique	Corpous	Accuracy
Back Off	Tested on a set of 10 highly polysemous Hindi words	65%
Deleted Interpolation	Tested on a set of 10 highly polysemous Hindi words	67%

The results with Deleted Interpolation method are better because it calculate the trigram probability by mixing together the unigram, bigram, and trigram probabilities. The above N-gram Smoothing model has been implemented using Perl programming language. The encoding used for text in Unicode. The accuracy of Word Sense Dis-ambuigator depends upon the number of entries in database and size of N-gram used. Database can be extended to include more entries to improve the accuracy.

V. CONCLUSION AND FUTURE SCOPE

WSD is the task of assigning a meaning to an ambiguous word given the context in which it occurs. There are various approaches used for Word Sense Disambiguation like Machine learning approaches and dictionary based approach. Each approach do have its own requirements for implementation. Due to very availability of resources we have chooses to work on dictionary based approaches. After studying number of works done by various researches in the area, we have developed new algorithm for disambiguating the Hindi words and the accuracy comes out to be 60% to 70% with Deleted Interpolation method and 50% to 60 with Back Off method depending upon the input sentence. Now, as future work, database can be extended to include more entries to improve the accuracy.

REFERENCES

[1] Francois Raphael and Lison Pierre. .May 04, 2005, *Probabilistic Language Modeling with N-grams*, Artificial Intelligence Seminar,pp 1-43

[2] Josan Gurpreet Singh and Lehal Gurpreet Singh. *Size of N for Word Sense Disambiguation using N gram model for Punjabi Language*, Yadwindra College of Engg., Talwandi Sabo,pp 1-11

[3] Daniel Jurafsky and James H. Martin. Speech and Language Processing: *An introduction to natural language processing ,computational linguistics, and speech recognition*,pp ,189-230

[4] Marti U.V. & Bunke H. 2001, *On the influence of vocabulary size and language models in unconstrained handwritten textrecognition*, Proc. 6th Int. Conference on Document Analysisand Recognition, pp 260 – 265.

[5] Sinha Manish et. al. 2004. *Hindi Word Sense Disambiguation*, International Symposium on Machine Translation, NLP& Translation Support Systems, Delhi, India, pp 230-237.

[6]https://en.wikipedia.org/wiki/Natural_language_processing Accessed on 27/3/2014

[7] http://en.wikipedia.org/wiki/N-gram accessed on Accessed on 27/3/2014