A Java Implementation of an Extended Word Alignment Algorithm Based on the IBM Models

G. Chinnappa and Anil Kumar Singh

Language Technologies Research Centre
International Institute of Information Technology
Hyderabad, India
guggilla@students.iiit.net, anil@research.iiit.net

Abstract. In recent years statistical word alignment models have been widely used for various Natural Language Processing (NLP) problems. In this paper we describe a platform independent and object oriented implementation (in Java) of a word alignment algorithm. This algorithm is based on the first three IBM models. This is an ongoing work in which we are trying to explore the possible enhancements to the IBM models, especially for related languages like the Indian languages. We have been able to improve the performance by introducing a similarity measure (Dice coefficient), using a list of cognates and morph analyzer. Use of information about cognates is especially relevant for Indian languages because these languages have a lot of borrowed and inherited words which are common to more than one language. For our experiments on English-Hindi word alignment, we also tried to use a bilingual dictionary to bootstrap the Expectation Maximization (EM) algorithm. After training on 7399 sentence aligned sentences, we compared the results with GIZA++, an existing word alignment tool. The results indicate that though the performance of our word aligner is lower than that of GIZA++, it can be improved by adding some techniques like smoothing to take care of the data sparsity problem. We are also working on further improvements using morphological information and a better similarity measure etc. This word alignment tool is in the form of an API and is being developed as part of Sanchay, (a collection of tools and APIs for NLP with focus on Indian languages).

Keywords: Word alignment, statistical machine translation, IBM models, Java, Indian languages, cognates, Sanchay.

1. Introduction

Bilingual word alignment is the first step of most current approaches to Statistical Machine Translation or SMT [3]. Most of the SMT systems usually have two stages. The first stage is called language modeling. One simple and very old but still quite useful approach for language modeling is *n*-gram modeling. Separate language models are built for the source language (SL) and the target language (TL). For this stage, monolingual corpora of the SL and the TL are required. The second stage is

called translation modeling and it includes the step of finding the word alignments induced over a sentence aligned bilingual (parallel) corpus. This paper deals with the step of word alignment, which is sometimes extended to phrase alignment.

The paper is structured as follows. In Section-1.1 we discuss how generative models can still be useful, especially for resource scarce languages. Section-1.1 briefly describes the extensions added to the IBM models. In Section-1.3 we relate IBM models and possible extensions to them with Indian languages. Section-2 describes some related work. Section-3 compares our implementation with GIZA++. Some of the problems of word alignment are discussed in Section-4. Word alignment parameters (for IBM models) are described in Section-5. The modified EM algorithm is presented in Section-6. Data and methodology for evaluation are presented in Section-7. In Section-8, we present some experimental results. In Section-9, we comment on the results. Finally, Section-10 lists some conclusions and some of the planned work for future.

1.1 Generative Models could still be Useful

Most current SMT systems [19, 11] use a generative model for word alignment such as the one implemented in the freely available tool GIZA++ [16]. GIZA++ is an implementation of the IBM alignment models [2]. These models treat word alignment as a hidden process, and maximize the probability of the observed *(e, f)* sentence pairs using the Expectation Maximization (EM) algorithm, where *e* and *f* are the source and the target sentences.

Recently a number of discriminative (as against generative) word alignment models have been proposed. However these early models are typically very complicated, with many proposing intractable problems which require heuristics for approximate inference [13, 17]. These discriminative models need to have more features to reach the performance of generative models. At the same time discriminative models needs annotated parallel corpora along with some linguistic resources. Discriminative models are more easily applicable to European and English languages (in practical terms) due to the availability of annotated corpora and linguistic resources to induce the features. But this is not true for Indian languages since most of the languages do not have annotated corpora and linguistic resources developed. Most of these languages do not have a good enough POS tagger to get the POS feature.

In fact, some more recent discriminative models are actually using the results obtained from IBM models as separate features and have achieved an increase in performance. This shows that, in spite of their limitations, generative models are still useful for a variety of reasons and attempts to improve them could be a fruitful exercise.

1.2 Extending the IBM Models

It is in this context that this paper presents an extended generative method based on the IBM word alignment models. One of the extensions is using a statistical similarity

measure along with the IBM translation parameters to reduce the number of inappropriate alignments. Currently we use the Dice coefficient [5] as the similarity measure to boost the Viterbi alignment so that the search time is reduced and precision also increases. Dice coefficient is calculated on the fly as translation parameters are calculated, which reduces the complication of the models as compared to the case when it is calculated while training. Even though the Dice coefficient selects word pairs from the parallel corpus which are more likely to be valid alignment, some highly frequent words such as 'the' and 'of' leads to lower alignment accuracy. This problem can be solved if we combine the Dice coefficient value of a particular word pair with value obtained from Viterbi alignment based on the original IBM models.

1.3 IBM Models for Indian Languages

Since most of the Indian languages have common words which are either borrowed or inherited [22, 23]. In this paper we will call words of both the categories 'cognates'. For better word alignment of text in Indian languages, information about cognates is certainly needed. Note that cognates are a useful source of information even when aligning English with Indian languages as Indian languages have borrowed a large number of words from English. The cognate list used by us was prepared by using a method based on the Computational Phonetic Model of Scripts or CPMS [21] which can be used for cognate identification [22, 23]. We added the cognates list to the bilingual dictionary (for English-Hindi) and we used this modified dictionary for the initialization of EM algorithm. We also performed a preliminary evaluation on Hindi-Bengali using the extension mentioned above (except bilingual dictionary).

Here is an example of word alignment between an English sentence and a Hindi sentence. The two parallel sentences have two word pairs which are cognates:

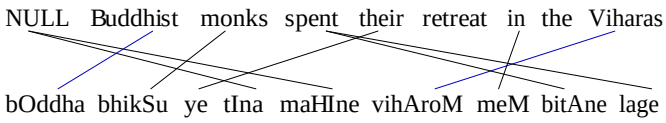


Fig. 1. An example of an alignment between an English-Hindi sentence pair, blue links indicates alignment of cognates

2. Related Work

In 1991, Gale and Church [9] introduced the idea of using measures of association for finding translations of words based on information in parallel text. They begin by carrying out sentence alignment, which is the problem of determining which sentences are translations of each other. In fact this is a much simpler problem than finding the translations of words, since long sentences in one language tend to

translate as long sentences in another language, and the order in which sentences appear doesn't usually change radically in a translation.

The original K-vec algorithm proposed by Fung and Church [7] works only for parallel corpus and makes use of the word position and frequency feature to find word correspondences. K-vec uses tests of association as a similarity measure, while the 1995 approach of Fung [6] uses Euclidean distance. Like K-vec this approach is also language independent and works for different language pairs. Fung and Yee [8] also proposed an IR approach for translating new words from non-parallel comparable texts.

Ittycheriah and Roukos [10] proposed a maximum entropy word aligner for Arabic-English machine translation. Malouf [14] compared several algorithms for maximum entropy parameter estimation. Martin et al. [15] have discussed word alignment for languages with scarce resources.

Most current SMT systems [19, 11] use a generative model for word alignment such as the freely available GIZA++ [18], which is an implementation of the IBM word alignment models [2] in C++. These models treat word alignment as a hidden process, and maximize the probability of the observed sentence pairs using the expectation maximization (EM) algorithm.

Recently, a number of discriminative word alignment models have been proposed. Taskar et al. [25] presented a word matching model for discriminative alignment which they were able to achieve optimally.

Liu et al. [13] also develop a log-linear model, based on IBM model-3. They train model-3 using GIZA++, and then use the model-3 score of a possible alignment as a feature value in a discriminatively trained log-linear model, along with features incorporating part-of-speech information, and whether the aligned words are given as translations in a bilingual dictionary. The log-linear model is trained by standard maximum-entropy methods.

Moore et al. [17] proposed a discriminative framework for bilingual word alignment. In 2006, Phil Blunsom and Trevor Cohn [1] proposed discriminative word alignment with Conditional Random Fields or CRF [12]. CRF has been used for many NLP problems like shallow parsing [20]. They used IBM model-4 results as a separate feature in their model. Sriram Venkatapathy and Aravind K. Joshi [24] proposed a generic discriminative re-ranking approach for word alignment which allows us to make use of structural features effectively. They also used IBM model-4 results as a separate feature in their method.

3. Comparison of Implementation with GIZA++

The difference between GIZA++ and our implementation is that we are using only the first three models and we have not implemented various techniques for optimization etc. (at least so far). However, we also use the Dice coefficient, a similarity measure, and combine it with the Viterbi approximation to EM. From practical point of view, our implementation is also object oriented, but it is implemented in Java, which makes it platform independent. The implementation is in the form of an API so that it can be used by other programmers. Moreover, since it will be a part of the open

source Sanchay (a collection of tools and APIs for NLP, especially for Indian languages), other researchers will be able modify and extend it for developing new algorithms which use the IBM models in some way. Like GIZA++, our implementation is also language independent, perhaps more so because Java allows the text in various languages and encodings to be processed more easily.

4. Problems in Word Alignment

The initial assumption for word alignment is that we have a sentence aligned parallel corpus of two languages. Now, given a parallel sentence pair, we can link (align) words that are translations of one another. There may be a large number of possible alignments, but we need to find the best alignment as shown below:

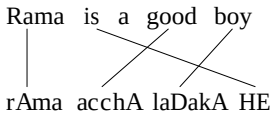


Fig. 2. An example of English-Hindi alignment

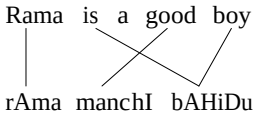


Fig. 3. An example of English-Telugu alignment

In approaches based on IBM models, the problem of word alignment is divided into several different problems. The first problem is to find the most likely translations of an SL word, irrespective of positions. This part is taken care of by the **translation model**. The model alone has many applications. For example, since this model gives probable of word translations, we can use this model to make the task of building a bilingual dictionary easier. The second problem is to align positions in the SL sentence with positions in the TL sentence. This problem is addressed by the **distortion model**. It takes care of the differences in word orders of the two languages. The third problem is to find out how many TL words are generated by one SL word. Note that an SL word may sometimes generate no TL word, or a TL word may be generated by no SL word (**NULL insertion**). The **fertility model** is supposed to account for this. The first three models corresponding to these problems form the core of the IBM model based generative SMT. Examples of these are shown in Figure-4.

Unlike European languages, most of the Indian languages are morphologically rich and have the feature of compounding, thereby making the problem different in terms of SMT. When we are trying to align two European languages, we are much more likely to get one-to-one alignments, but when at least one of the languages is an

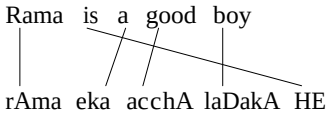
Indian language, this is less likely. In other words, the problem is much harder for the fertility model. One-to-many or many-to-one translations are much more likely and so is NULL insertion.

Since English is an SVO language and Indian languages are SOV with respect to the word order, alignment of word positions may also be more difficult when one language in an Indian language and the other is a European language, like English. This will make the task of the distortion model harder. But this will not be a problem if both the languages are Indian languages.

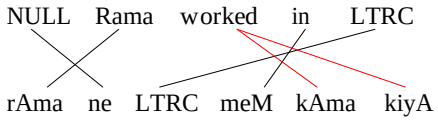
Apart from compounding, tense, aspect and modality (TAM) of Indian language verbs also are a cause of errors in alignment. This is because the TAM information is distributed over several words, which causes problems for the fertility model. This is, in fact, one of major factors in reducing the alignment accuracy.

However, there are some aspects which, if used properly, may allow us to get good accuracy with approached bases on IBM models. As mentioned earlier, Indian languages have a lot of borrowed and inherited words which are common to more than one language. Using a list of cognates or aligning cognates on the fly using better techniques like the ones based on the CPMS [21, 22], we can increase the accuracy of alignment. If a bilingual dictionary is available, we can use that to initialize the EM algorithm.

Ex-1. Translation (one to one alignment):



Ex-2. Distortion (word order) and NULL insertion (‘spurious’ words):



Ex-3. Fertility:

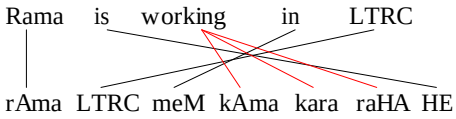


Fig. 4. Problems in word alignment which the first three IBM models try to solve. The red links indicate the fertility problem due to compounding in Indian languages.

5. Parameters for Word Alignment

In this section we briefly describe the problem of word alignment in a formal notation. The problem of SMT is: Given a source sentence how do we get (generate) the corresponding target sentence as a translation?

To develop a statistical machine translation system from Hindi to English, we need to have a model ***p(e | h)*** which estimates the conditional probability of any English sentence e given the Hindi sentence ***h***. We need to use the parallel corpus to set the parameters. We can find ***p(e | h)*** using the Bayes' theorem:

$$p(e \mid h) = \frac{p(e) \times p(h \mid e)}{p(h)} \quad (1)$$

Usually, a statistical machine translation system consists of two components: the **language model *p(e)*** and the **translation model *p(h | e)***. The **language model *p(e)*** could be a bigram or trigram model, estimated from any data, i.e. parallel corpus is not needed to estimate the parameters. But the **translation model *p(h | e)*** has to be trained from a sentence aligned English-Hindi parallel corpus. The probability ***p(h | e)*** is the probability that the Hindi word ***h*** is a translation of the English word ***e***.

Since we are interested here in word alignment, we will focus only on computing ***p(h | e),*** i.e., the translation parameter, from parallel corpus. Note that there is some terminological ambiguity here in the sense that this bigger translation model itself consists of three models in the IBM model approach as described earlier: the translation model, the distortion model and the fertility model. We will consider these three (sub-)models in this section.

The translation model uses the simple idea of co-occurrence of ***e*** and ***h*** in the parallel corpus: If ***e*** and ***h*** tend to co-occur in parallel sentence pairs, they are likely to be translations of one another. We can also say that ***e*** and ***h*** are likely to be distributionally similar.

We can calculate the translation probabilities from the word alignment probabilities. But we can also calculate word alignments from the translation probabilities. This chicken-and-egg like recursive situation is where the Expectation Maximization (EM) algorithm comes in.

5.1 Model-1: Translation Parameter

Given an English sentence $e_1 \ldots e_l$ and a target sentence $h_1 \ldots h_m$ from the parallel corpus, we want to find out the best alignment ***a***, where ***a*** is a vector $a = \{a_j, a_{j+1}, a_{j+2}, \ldots, a_m\}_{j=1 \text{ to } m}$. An alignment ***a*** specifies which English word each Hindi word was generated from. We add a spurious NULL word to the English sentence at position 0. Thus, there are $(l + 1)^m$ possible alignments.

Since in model-1 word positions are not considered, the probability of alignment of any two positions is a constant equal to $\frac{1}{(l+1)^m}$ for a particular sentence pair. The probability of generating a Hindi word given an English word is given as:

$$p(h \mid a, e) = \prod_{j=1}^{m} T(h_j \mid e_{a_j}) \quad \textbf{(2)}$$

Therefore, from Bayes' rule:

$$p(h, a \mid e) = \frac{1}{(l+1)^m} \times \prod_{j=1}^{m} T(h_j \mid e_{a_j}) \quad \textbf{(3)}$$

where ***p(h, a | e)*** is the probability of generating a target word and an alignment given an English word.

5.2 Model-2: Distortion or Alignment Parameter

Given source and target sentence lengths ***l*** and ***m***, probability that ***jth*** target word is connected to ***ith*** source word, the distortion probability is given as ***D(i | j, l , m)***. Now, the probability of an alignment ***a***, given the source word and the lengths of the source and target sentences is:

$$p(a \mid e, l, m) = \prod_{j=1}^{m} D(a_j \mid j, l, m) \quad \textbf{(4)}$$

where $\boldsymbol{a = \{a_1, \ldots, a_m}$ and $\boldsymbol{a_j}$ is the position in source sentence that aligns to the jth position in the target sentence, i.e., $\boldsymbol{a_j}$ is ***i***.

Now the probability of generating a target word with alignment ***a***, given the source word and the lengths of the source and target sentences can be calculated as:

$$p(h, a \mid e, l, m) = \prod_{j=1}^{m} D(a_j \mid j, l, m) \times T(h_j \mid e_{a_j}) \quad \textbf{(5)}$$

5.3 Model-3: Fertility Parameter

We can generate the target sentence from English sentence with the probability ***p(h, a | e)***. In the third model, this probability is calculated using a new parameter called fertility Φ, where ***F(e |*** Φ***)*** = probability that ***e*** is aligned with Φ target words. Model-3 uses translation and distortion probabilities from model-2. Fertility probabilities are generated from the model-2 as a separate step. And we uniformly assign the reverse distortion probabilities ***R*** for model-3. The final translation parameter is generated using four these parameters after learning from the parallel corpus:

$$p(h,a \mid e) = \prod_{i=1}^{l} F(\Phi_i \mid e_i) \times \prod_{j=1}^{\Phi_i} D(i \mid j,l,m) \times R(j \mid i,l,m) \times T(h_j \mid e_i) \qquad (6)$$

6. Expectation Maximization (EM) Algorithm for Training

For calculating the parameters mentioned above (translation, distortion and fertility) we use a generative algorithm called Expectation Maximization (EM) for training. The EM algorithm guarantees an increase in likelihood of the model in each iteration, i.e., it is guaranteed to converge to a maximum likelihood estimate.

A set of sentence aligned parallel corpus is used as the training data. Let the number of sentence pairs in the training data be ***N*** and the lengths of the source and target sentences be ***l*** and ***m***, respectively. The translation parameter ***T*** is learned during training using expected translation counts t_c. After training, word alignment ***a*** is induced from the translation parameter. Let the number of iterations during training be ***n***. Then, the iterative EM algorithm corresponding to the translation problem can be described as:

Step-1: Collect all word types from the source and target corpora. For each source word ***e*** collect all target words ***h*** that co-occur at least once with ***e***.

Step-2: Initialize the translation parameter uniformly (uniform probability distribution), i.e., any target word probably can be the translation of a source word ***e***.

$$T(h \mid e) = 1/(\text{ number of co-occurring target words}) \qquad (7)$$

Step-3: Iteratively refine the translation probabilities until values are good enough

```
for n iterations do
  initialize the expected translation count t_c to 0
  for each sentence pair (e, h) of lengths l, m do
    update the expected translation count
```

```
for j=1 to m do
  set total to 0
    for i=1 to l do
      total += T(h_j|e_i)
     for i=1 to l do
       t_c(h_j|e_i) += T(h_j|e_i)/total
      end for
    end for
  end for
end for
re-estimate the translation parameter values
for each source word e do
  set total to 0
    for each target word f do
      total += t_c(h_j|e_i)
        for each target word f do
          calculate T(h_j|e_i)= t_c(h_j|e_i)/total
        end for
      end for
    end for
  end for
```

After the training we will have translation probability values for source and target words. Since, in IBM model theory, $T(h_j \mid e_i)$ is assumed to be independent from $T(h_j' \mid e_i')$, we can find the best alignment by looking at the individual translation probability values. The best alignment can be calculated in a quadratic number of steps equal to $(l+1) \times m$.

The best (Viterbi) alignment $\boldsymbol{a}$ is now given by:

$$a_{j=1}^{m}[i, j] = \arg\max(T(h_j \mid e_{a_j})) \quad (8)$$

The above EM algorithm is applied to learn the translation parameters for finding the correct word alignment. Similarly we can use the EM algorithm to solve the distortion (word order) and fertility problems by learning corresponding distortion and fertility parameters.

Distortion parameters are calculated according to the source and target sentence lengths ***l***, **m**. To get the distortion parameters we use the corresponding expected translation counts from model-1 for the training in model-2. After getting the translation and distortion parameters from the model-2 training, the best (Viterbi) alignment can be calculated as follows:

$$a_{j=1}^{m}[i, j, l, m] = \arg\max(D_i \mid j, l, m) \times T(h_j \mid e_i)) \quad (\mathbf{9})$$

This can be calculated in ***(l+1)×m*** steps.

In the same way, the EM algorithm is trained to learn the fertility parameter in the model-3. For training, we use four parameters. Translation and distortion parameters are taken from the model-2. Initial fertility parameter and reverse distortion parameter for model-3 are generated from model-2 to model-3 as an intermediate step. Thus, to train the model, we use expected fertility, distortion, reverse distortion and translation counts. In each iteration, these counts are estimated and re-estimated (maximized).

After certain number of iterations (set to 5 in our experiments) we will get the final four parameters. The best (Viterbi) alignment can be calculated as:

$$\begin{aligned} a_{j=1}^{m}[i, j, l, m] &= \arg\max(D_i \mid j, l, m) \times T(h_j \mid e_i)) \\ &\times D_{rev}(j \mid i, l, m) \times F(\Phi_i \mid e_i) \end{aligned} \quad (\mathbf{10})$$

6.1 Dice coefficient and Alignment search Optimization

To get better alignments from the above parameters we use the Dice coefficient to eliminate invalid alignments, i.e., the alignments whose Dice coefficient value is low. The Dice co-efficient is defined as:

$$\text{Dice coefficient (e, h)} = \frac{2 \times \text{count(e, h)}}{\text{count(e)} + \text{count(h)}} \quad (\mathbf{11})$$

We use the following modified Viterbi value to eliminate the low probability alignments, thereby increasing the word alignment accuracy:

Modified Viterbi value = alignment value [i, j, l, m] + Dice coefficient (e_i, h_j) (**12**)

In other words, the Dice coefficient can be used as another parameter to get the best word-alignment in unsupervised learning.

6.2 Morph Analyzer for Better Word Co-occurrence Learning

Since one word can have many forms, it may be possible to improve the learning of co-occurrence information if we lemmatize the words in SL and TL corpora using a morphological analyzer. Such preprocessing for obtaining the root forms can be extremely useful, particularly for Indian languages because they are morphologically rich. We also present the results after using the morph analyzer as shown in table-4. There was a significant improvement in alignment accuracy in this case.

The examples below show the lemmatized forms of English and Hindi words after using morph analyzers for removing suffixes:

Inernationalization	?	International	+	ization
		(root)		(suffix)
antharjAthIyaakaranaM	?	antharjAthIya	+	karanaM
		(root)		(suffix)
went	?	go	+	Past Tense
		(root)	+	morph
gayA	?	jA	+	wA/ wI
		(root)	+	morph

6.3 Dictionary and Cognates List for Initialization of the EM algorithm

We have used Shabdaanjali dictionary (bilingual English-Hindi dictionary) which consists of 27,000 word to word translations. We have the dictionary (in the English-Hindi experiments) along with the cognates list as an initialization step for the EM algorithm, to improve the word alignment accuracy. We generated cognates list by using a phonetic mapping technique [21, 22] from the ERDC (C-DAC, NOIDA) English to Hindi parallel corpora of 50,000 sentences.

7. Data and Methodology for Evaluation

We used the English-Hindi corpus (50,000 sentence pairs) to obtain the similarity measure the similarity of SL and TL words given by the Dice co-efficient, which we have used in our model. This data was also used to train GIZA++. We compare our results with those given by GIZA++. For evaluation, we used 2,000 sentence pairs as the test data. We tested only on sentences which were at least 5 words long.

We also evaluated our models using the English-French data from the bilingual word alignment workshop held at HLT-NAACL 2003 [16]. This data had 447 manually word-aligned sentence pairs. We report the performance of our alignment models in terms of precision, recall, and alignment error rate (AER) defined as:

Precision = $|A \cap C| / |A|$

Recall = |A ∩ C| / |C

$$\textbf{F-measure} = \frac{2 \times precision \times recall}{precision + recall}$$

AER = 1 – F-measure

In these definitions, C denotes the set of correct manually annotated alignments and *A* denotes the set of alignments produced by the method under test. Following standard practice, we take AER, which is derived from F-measure, as the primary evaluation metric that we are attempting to optimize.

8. Experimental Results

Experiment-1

This was a controlled experimented in which 100 sentence pairs were handcrafted so that there was no data sparsity problem, i.e., the data was designed to be good to learn from. The experiment was mainly meant to verify that our implementation is correct.

Training data: 100 sentences of clean English-Hindi parallel corpora
Testing data: 50 random (manually annotated) sentences from training data

	Our Model	**GIZA++**
Precision	42.07	37.83
Recall	49.25	47.14
F-Measure	45.38	41.97
AER	54.62	58.12

Table 1. Results of experiment-1 (controlled experiment)

Experiment-2

Data: 447 English-French sentences from the NAACL-2003 shared task data

	Our Model	**GIZA++**
Precision	38.16	58.97
Recall	16.36	25.29
F-Measure	22.90	35.35
AER	77.10	64.65

Table 2. Results of eperiment-2 (English-French)

Experiment-3

Training data: 2,000 sentences
Testing data: 100 sentences

	Our Model	GIZA++	Our Model with Cognate List
Precision	27.55	42.73	31.50
Recall	27.13	50.97	33.23
F-Measure	27.43	46.49	32.34
AER	72.66	53.51	67.66

Table 3. Results of eperiment-3 (Hindi-Bengali with cognates).

Experiment-4

Trained on: 7399 sentences
Tested on: 1600 sentences
English-Hindi vocabulary size: 50,000 approximately
Number of cognates: 24,000 approximately

Experimental Setups:

S1 is Our Model
S2 is Our Model + Dice Co-efficient
S3 is Our Model + Dice Co-efficient + Morph
S4 is Our Model + Dice Co-efficient + Morph + Cognates
S5 is Our Model + Dice Co-efficient + Morph + Cognates + Bilingual Dictionary

	S1	S2	S3	S4	S5	GIZA++
Precision	15.79	18.03	24.22	24.97	28.99	36.91
Recall	20.15	23.12	27.58	28.42	32.39	38.57
F-measure	17.70	20.26	25.79	26.58	**30.59**	**37.72**
AER	82.30	79.73	74.21	73.42	**69.41**	**62.38**

Table 4. Results of Experiment-4 (English-Hindi, using the Dice co-efficient, cognates, morphological analyzer and a bilingual dictionary).

9. Some Comments about the Results

As can be seen from the results above, the performance of our implementation is lower in all cases (except in the first experiment) than that of GIZA++. This is true even when we use a cognate list. However, two things may be noted here.

First, our implementation is meant to be a starting point for developing new algorithms, either by ourselves or by others. The performance right now was bound to be lower because GIZA++ uses model-4 also, along with many other optimizations and smoothing techniques. The results of the first (controlled) experiment indicate that our implementation works well when the data is good for learning, i.e., when

there is less data sparsity. This implies that we need to add some techniques to take care of the data sparsity problem.

Second, it is quite clear that there is a noticeable increase in performance when we use a cognate list. The fact that the performance was higher in the first experiment shows that our implementation is indeed working, but as the results for other experiments show, GIZA++ is better able to learn from larger data and is able to overcome data sparsity problem to a greater extent.

Since this is an ongoing work, we will be adding several new features, which are likely to increase the performance significantly. Even with performance comparable to or slightly lower than GIZA++, our implementation will be useful to community of NLP researchers, especially those who are working on Indian languages.

The results for the third experiment (Hindi-Bengali) show our method works better for when both the languages are Indian languages. This is mainly because of the common words as mentioned earlier as we have used a cognate list to take care of borrowed and inherited words.

10. Conclusions and Future Work

We have implemented the first three IBM models in JAVA in the form of an API. We can issue a query to get any parameter or results from the IBM models. We also added the Dice coefficient similarity measure as a parameter to the EM algorithm to improve the word alignment accuracy. We briefly described the first three IBM models. We conducted four experiments, out of which the first one was a controlled experiment. The results we achieved were lower than those obtained with GIZA++ in all the experiments except the first. We discussed the reasons and implications of these results. For example, GIZA++ uses the first four IBM models while we use only the first three. Moreover, GIZA++ also uses optimization and smoothing techniques. Also, our implementation is meant to be a starting point for developing an API for NLP researchers willing to use the IBM models in some way, rather than just an alignment tool. However, the results for Hindi-Bengali were comparatively much better, which shows that using a cognate list is increasing the performance significantly. In future, we will work on removing the limitations of our implementation by including the model-4, using optimization and smoothing techniques and adding some more features for Indian languages. Another important work for the future is to take care of the compounding problem in Indian languages, which is a major reason for low performance with Indian languages.

References

1. Phil Blunsom and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL (ACL 2006). Pages 65-72. Sydney, Australia.

2. P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L.Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263–311.
3. C. Callison-Burch, D. Talbot, and M. Osborne. 2004. Statistical machine translation with word- and sentence-aligned parallel corpora. In Proceedings of ACL, pages 175–182, Barcelona, Spain, July.
4. S. Chen and R. Rosenfeld. 1999. A survey of smoothing techniques for maximum entropy models. IEEE Transactions on Speech and Audio Processing, 8(1):37–50.
5. L. R. Dice. 1945. Measures of the amount of ecologic association between species. Journal of Ecology, 26:297–302.
6. Pascale Fung. 1995. "Compiling Bilingual Lexicon Entries from a Non-Parallel English-Chinese Corpus", In Third Annual Workshop on Very Large Corpora, Boston, Massachusetts: Jun. 1995. Pages 173-183.
7. Pascale Fung and Kenneth Ward Church. 1994. K-vec: a new approach for aligning parallel texts. In Proceedings of the 15th conference on Computational linguistics. Pages 1096-1102. Kyoto, Japan.
8. P. Fung and L. Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*. Pages 414–420.
9. W. Gale and K. Church. Identifying word correspondences in parallel texts. 1991. In *Proceedings of theFourth DARPA Speech and Natural Language Workshop*. Pages 152–157, Pacific Grove, CA,.
10. Ittycheriah and S. Roukos. 2005. A maximum entropy word aligner for Arabic-English machine translation. In Proceedings of HLT-EMNLP. Vancouver, Canada. Pages 89–96.
11. P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase based translation. In Proceedings of HLT-NAACL. Edmonton, Canada. Pages 81–88.
12. J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In Proceedings of ICML. Williamstown, USA. Pages 282–289.
13. Y. Liu, Q. Liu, and S. Lin. 2005. Log-linear models for word alignment. In Proceedings of ACL, AnnArbor, USA. Pages 459–466.
14. R. Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In Proceedings of CoNLL. Taipei, Taiwan. Pages 49–55.
15. J. Martin, R. Mihalcea, and T. Pedersen. 2005. Word alignment for languages with scarce resources. In Proceedings of the ACL Workshop on Building and Using Parallel Texts. Ann Arbor, USA. Pages 65–74.
16. R. Mihalcea and T. Pedersen. 2003. An evaluation exercise for word alignment. In Proceedings of HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond. Edmonton, Canada. Pages 1–6.
17. R. C. Moore. 2005. A discriminative framework for bilingual word alignment. In Proceedings of HLT-EMNLP. Vancouver, Canada. Pages 81–88.
18. F. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. Computational Linguistics, 29(1):19–52.
19. F. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. Computational Linguistics, 30(4):417–449.
20. F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In Proceedings of HLT-NAACL. Edmonton, Canada. Pages 213–220.
21. Anil Kumar Singh. 2006. A Computational Phonetic Model for Indian Language Scripts. Constraints on Spelling Changes: Fifth International Workshop on Writing Systems. Nijmegen, The Netherlands.

22. Anil Kumar Singh and Harshit Surana. 2007a, Study of Cognates among South Asian Languages for the Purpose of Building Lexical Resources. In Proceedings of National Seminar on Creation of Lexical Resources for Indian Language Computing and Processing. Mumbai, India.
23. Anil Kumar Singh and Harshit Surana. 2007b . Can Corpus Based Measures be Used for Comparative Study of Languages? In Proceedings of the Ninth Meeting of the ACL Special Interest Group in Computational Morphology and Phonology. Prague, 2007. ACL.
24. Sriram Venkatapathy and Aravind Joshi. 2007. Discriminative word alignment by learning the alignment structure and syntactic divergence between a language pair. In Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation. Rochester, New York. Pages 49-56.
25. Taskar, S. Lacoste-Julien, and D. Klein. 2005. A discriminative matching approach to word alignment. In Proceedings of HLT-EMNLP. Vancouver, Canada. Pages 73–80.