Joint prediction of word alignment and alignment types for statistical machine translation

by

Te Bu

B.Sc., Simon Fraser University, 2013
B.Sc., Zhejiang University, 2013

Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

Approval

Name:	Te Bu
Degree:	Master of Science (Computing Science)
Title:	*Joint prediction of word alignment and alignment types for statistical machine translation*
Examining Committee:	**Dr. William Nick Sumner** (chair) Assistant Professor

Dr. Anoop Sarkar
Senior Supervisor
Professor

Dr. Fred Popowich
Supervisor
Professor

Dr. Greg Mori
Internal Examiner
Professor
Computing Science Department
Simon Fraser University

Date Defended:	17 November 2015

Abstract

Learning word alignments between parallel sentence pairs is an important task in Statistical Machine Translation. Existing models for word alignment have assumed that word alignment links are untyped. In this work, we propose new machine learning models that use linguistically informed link types to enrich word alignments. We use 11 different alignment link types based on annotated data released by the Linguistics Data Consortium. We first provide a solution to the sub-problem of alignment type prediction given an aligned word pair and then propose two different models to simultaneously predict word alignment and alignment types. Our experimental results show that we can recover alignment link types with an F-score of 81.4%. Our joint model improves the word alignment F-score by 4.6% over a baseline that does not use typed alignment links. We expect typed word alignments to benefit SMT and other NLP tasks that rely on word alignments.

Keywords: Statistical Machine Translation; Word Alignment; Alignment Type; Probabilistic Models

Dedication

To my family!

Acknowledgements

First, I would like to thank my family for their love and support throughout my entire study. My parents give me complete freedom on deciding what to study and where to study and always stand by me no matter what decision I made. Without them I can never go this far and complete my Master's degree.

Second, I want to say thank you to my senior supervisor Anoop Sarkar who has been very patient and tolerant to me during this research work. He is the one that led me into the door of scientific research and his creative and constructive insights often inspired me. It has been a good time working with him. Special thanks also go to my supervisor Fred Popowich, who has been giving me many useful advice and feedback on this work.

Third, I want to thank my lab-mates Jasneet, Ramtin, Bradley, Zhelun, Lydia, Hassan, Mehdi, Vivian, Golnar, Maryam, Anahita, and Ann for the good time spent with them. I would also like to thank my other friends who make my life in Vancouver full of happiness and for their help and support for my work.

Learning is a lifelong journey. This is not the end but the beginning.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Description

Statistical Machine Translation (SMT) is the task of translating a source sentence in one language into the target sentence in another language. A typical SMT pipeline usually starts from learning word alignments from a parallel corpus and then use these word alignments to establish word translation tables. For a phrase based SMT model, word translations can be further combined into phrases and used in a log linear discriminative model over phrase translations.

As a crucial intermediate stage of SMT, finding the word alignments between a parallel corpus is an interesting and long-lasting research topic. The accuracy of word alignment will directly affect the quality of the following machine translation process. Traditional models like IBM models (Brown et al. 1993) use unsupervised algorithms to learn the hidden word relations between parallel text. With the increasing availability of manually word-aligned data, supervised approaches become a feasible alternative to unsupervised approaches. These models, like the discriminative word alignment model (Taskar et al. 2005) and Berkeley Aligner (Percy et al. 2006), usually use linguistic information such as part-of-speech tags and parse trees to yield better word alignments.

However, all previous work has assumed that word alignments are untyped, which means there's no difference between them. This is not really the fact if we take a close look from a linguistic perspective since the words aligned play different roles in a sentence and the reason to align them is different, too. In 2014, the LDC (Linguistic Data Consortium) released a new dataset which enriches word alignment with linguistic tags. These tags represent the types of the word alignment and cast light on building type-aware alignment models which hopefully can find better word alignment and also alignment types. An example of the enriched word alignment looks like Figure 1.1 below

In Figure 1.1, each tag on an alignment link is the type of this word alignment. For example the tag **SEM** stands for semantic alignment, which means the two aligned words

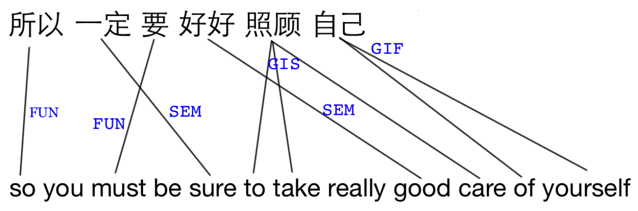

所以 一定 要 好好 照顾 自己
FUN
FUN
SEM
GIS
SEM
GIF
so you must be sure to take really good care of yourself

Figure 1.1: Example for enriched word alignment

are actually content words that carry equivalent rich lexical meaning, while the tag **FUN** stands for function alignment, which means the two aligned words carry less or ambiguous lexical meaning but instead serve to express grammatical relationships with other words in a sentence, like word *must*, *a*, and *in*. Throughout this work, we may interchangeably use the term **alignment type** and **tag**. They refer to the same thing.

With the alignment types enriched data at hand, we see the chances of finding maybe better quality word alignment and the alignment types at the same time. Thus, in this thesis, we also try to solve a joint problem, finding the word alignments between parallel text and predicting the corresponding alignment types at the same time.

1.2 Motivation

Empirical results have shown that supervised models can generally achieve better alignments than traditional unsupervised models when proper linguistic features are in good use. However, such linguistic features are usually part-of-speech tags, word stems and even parse information. The alignment types for word alignment as seen in the LDC data have not been used as a feature for finding better alignments. But it is definitely worth trying and looks very promising to explore the possibility of using such tags as a feature. We would expect the word alignment found in our joint task tends to be better than that found in a pure word alignment task. And if we can prove that such alignment types are useful, we can further include it in more complex machine translation system.

Moreover, those alignment types are by themselves very interesting. They can be used not only for finding better alignments but also for other Natural Language Processing (NLP) tasks that rely on word alignments such as projection of part of speech tags and dependency trees from a resource-rich language to a resource-poor language. From this perspective, we also want to be able to predict such alignment types while finding the alignments. Thus, we want to propose the new task of joint prediction of word alignment and alignment types.

1.3 Word Alignment Models

1.3.1 Words and Word Alignments

In natural language, the concept of word may vary for different languages. The concept of word is pretty clear in English because the writing system naturally separate the words with spaces. However, for some other languages where their writing system does not incorporate spaces, the boundary of a word can be ambiguous. The best example is Chinese, where a word can be just one character or multiple characters. The problem of word segmentation and the linguistic concept of word is not in the scope of this thesis so we don't discuss it in detail. We did all our experiments on English-Chinese parallel text, and we just use segmented Chinese where spaces are already inserted. Here we assume word to be the token separated by spaces, no matter how many characters it contains.

Word alignment is an important notion introduced in early statistical machine translation work (Brown et al. 1993). Given a sentence and its translation, the corresponding word pair, for example like English word *So* and Chinese word *所以* in Figure 1.2, are called a word alignment. We usually draw a link between two words to represent that they are aligned, as shown in the Figure below. And the same relationship can also be visualized by the alignment matrix in Table 1.1:

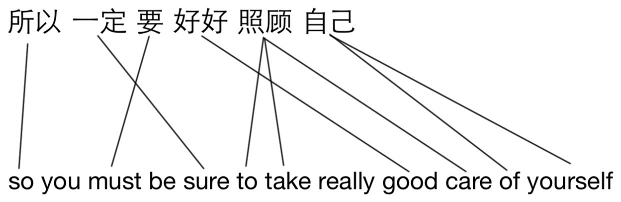

所以 一定 要 好好 照顾 自己
so you must be sure to take really good care of yourself

Figure 1.2: Example of traditional word alignment

Word alignment does not have to be one-to-one. This is fairly simple to understand since one word in one language does not necessarily translate into only one word in another language. It can be two, three, or even more. This is because of the lexical divergence between languages. Typically there are four types of word alignments: one-to-one, one-to-many, many-to-one, and many-to-many. Besides those types of word alignments, a word in one language can also be aligned to a **NULL** word in another language, which means it actually gets eliminated in another language and does not receive a specific translation.

Usually, we measure the quality of word alignments by comparing the experimental results with the gold alignments and calculate the **Alignment Error Rate**(AER). The gold alignments are produced by bilingual human experts who went through the parallel corpus and manually aligned the words. The alignment error rate is defined as below:

	所以	一定	要	好好	照顾	自己
so	■					
you						
must			■			
be						
sure		■				
to					■	
take					■	
really						
good				■		
care					■	
of						■
yourself						■

Table 1.1: Alignment Matrix Example

$$AER(S, P; A) = \frac{|A \cap S| + |A \cap P|}{|A| + |S|} \tag{1.1}$$

Where A stands for the experimental results, S stands for the sure alignments in the gold alignments, and P stands for possible alignments in the gold alignments. The lower the alignment error rate is, the better the word alignment quality is. However, in the LDC data we used for our experiment, there is no distinction between sure and possible alignments. And also In (Fraser and Marcu, 2007) they provide a compelling argument that F-score is a much better evaluation measure for word alignment when compared to alignment error rate (AER). Thus, instead we report the precision, recall and F-score on the test data to measure the quality of our output word alignments. Precision, recall and F-score are widely used measures in the field of information retrieval and NLP. **Precision**, also called positive predictive value, is the fraction of retrieved/generated results that are relevant/correct. And it is defined as:

$$Precision = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \tag{1.2}$$

Recall is the fraction of relevant/correct results that are retrieved/generated. And it is defined as:

$$Recall = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \tag{1.3}$$

And **F-score** is the harmonic mean of precision and recall:

$$F - score = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{1.4}$$

1.3.2 IBM Model 1 Word Alignment

When talking about the models to learn word alignment, we have to mention the classic IBM models. IBM models are a series of translation models developed by Brown et al. (1993). They are still being used currently. The IBM models were implemented into a toolkit called GIZA++. GIZA++ is open source and widely used for finding word alignments. Among those models, IBM model 1 is the most basic one and acts as the baseline of our experiments. Thus, here we offer some background knowledge of IBM model 1 including the formal definition, how it is applied to find word alignments between parallel text and its limitations.

Definition

Following the discussion in the widely used textbook Statistical Machine Translation (Philipp Koehn 2010), if two parallel sentences are given, $\mathbf{e}$ the target sentence and $\mathbf{f}$ the source sentence, what is the probability of translating $\mathbf{f}$ into $\mathbf{e}$? It is a little bit hard to imagine this problem directly, because the same sentence does not often appear as a whole in a text. Instead, usually what occurs many times is just part of the sentence, which we call a word. Thus, this problem can be broken down and largely simplified if we view it from a word perspective.

The probability distribution of a foreign word f translates into an English word e can be easily estimated by counting how many times they co-occur in the parallel text. For example, if we get the statistics in Table 1.2 below after going through the text, we can easily estimate the translation probability distribution.

Table 1.2: Example counts for different translations of the Chinese word 打

Translation of 打	Count
hit	7000
punch	1500
fight	800
strike	500
dozen	200

Table 1.2 basically tells us that among the 10000 times that the Chinese word 打 appears, it is translated into the English word hit for 7000 times. So the translation probability is

$7000/10000 = 0.7$ and the overall translation probability distribution can be represented as

$$p_c(e) = \begin{cases} 0.7 & \text{if } e = hit \\ 0.15 & \text{if } e = punch \\ 0.08 & \text{if } e = fight \\ 0.05 & \text{if } e = strike \\ 0.02 & \text{if } e = dozen \end{cases} \tag{1.5}$$

This type of estimation is also called maximum likelihood estimation since it maximizes the likelihood of the data. And the word translation here is also called lexical translation. Now we have gained the probability distributions for lexical translation, how can we estimate the probability of the whole sentence translation? Let's look at an example below, we have a Chinese sentence **c** and its corresponding translation **e**

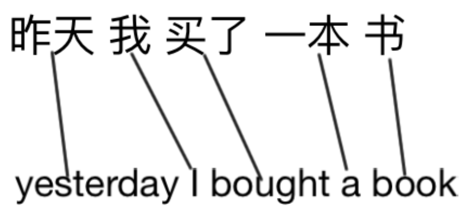


Figure 1.3: A Chinese sentence and its translation

The mapping here between the Chinese words and English words is called alignment. Mathematically, we define the alignment to be an alignment function **a**, which maps each English word at position i to its source Chinese word at position j:

$$a : j \to i \tag{1.6}$$

Thus, the alignments in our example above can be represented as:

$$a : \{1 \to 1, 2 \to 2, 3 \to 3, 4 \to 4, 5 \to 5\} \tag{1.7}$$

Equipped with the background of lexical translation and word alignment, now we are able to introduce IBM model 1, which is a lexical translation based model that produces a series of translations for a sentence, each with a different probability. In IBM model 1, the translation probability of a foreign sentence $\mathbf{f} = (f_1, ..., f_{l_f})$ of length l_f aligned to an English sentence $\mathbf{e} = (e_1, ..., e_{l_e})$ of length l_e with an alignment function a solely depends on the translation probability of each word, and it is defined as follows:

$$p(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j | f_{a(j)}) \tag{1.8}$$

Note that some people, including the original work of (Brown et al. 1993), use a slightly different equation where they model $p(\mathbf{f}|\mathbf{e})$ instead of $p(\mathbf{e}|\mathbf{f})$. Both of them are equally valid ways to model word alignment. Brown et al. use $p(\mathbf{f}|\mathbf{e})$ because they model alignment as part of the noisy channel model for translation and write $p(\mathbf{e}|\mathbf{f})$ as $p(\mathbf{e}) * p(\mathbf{f}|\mathbf{e})$ and use word alignment models to learn $p(\mathbf{f}|\mathbf{e})$ and a language model for $p(\mathbf{e})$. In contrast, statistical machine translation models that use a discriminative log-linear model to directly learn $p(\mathbf{e}|\mathbf{f})$ can simply use a word alignment model for $p(\mathbf{e}|\mathbf{f})$ as a source of feature functions in the log-linear model.

This equation tells us that the probability of translating a whole sentence is the product of the lexical translation probabilities. Since the mapping, i.e. alignment, between the two sentences is not unique, we need to normalize the right hand side of the equation. We assume that each type of alignment is equally likely, so the total number of ways to align the words between the two sentences should be $(l_f + 1)^{l_e}$. 1 is added because we include the special NULL token. ϵ is a constant used to make sure that $p(\mathbf{e}, a|\mathbf{f})$ is a proper distribution, which means to make the sum of all probabilities equal to 1.
Finally, we can apply the IBM model 1 to our previous example in Figure 1.3:

$$p(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{5^4} \times t(\textit{Yesterday}|\text{昨天}) \times t(\textit{I}|\text{我}) \times t(\textit{bought}|\text{买了}) \times t(\textit{a}|\text{一本}) \times t(\textit{book}|\text{书}) \tag{1.9}$$

EM for IBM Model 1

The sentence translation probability computed by IBM Model 1 relies on the lexical translation probabilities, which we assume we already have. However, in real world circumstances, usually we do not have such statistics. What we have is just parallel text. Thus, we need a way to learn those lexical translation probabilities from the parallel text. The unsupervised method adopted by Brown et al. (1993) is the expectation maximization algorithm. For the sake of brevity, we will use the commonly used abbreviation and call it the EM algorithm in the rest of the thesis.

In statistics, the EM algorithm is an iterative method used to find the maximum likelihood estimates of the parameters in statistical models. Such models often depend on latent variables. The EM algorithm iterates between an expectation (E) step, where the expected likelihood of the data is evaluated using the current parameters, and a maximization (M) step, which updates the estimates for the parameters. These new estimates for parameters are then used in the next E step.

In our case, we only have the parallel text. What we lack is the mapping between the words in a sentence pair, i.e. the alignment. Thus, the alignment is considered to be the latent variable in our model. If we had the word alignments marked up in our parallel text, we could easily estimate the parameters of our translation model, i.e. the lexical translation probabilities. On the other hand, if we had the parameters given to us, we could easily find the most likely alignment by searching for an alignment that maximizes the product of the lexical translation probabilities. Unfortunately, we got neither of them. Here is why EM algorithm comes in to help solve the problem.

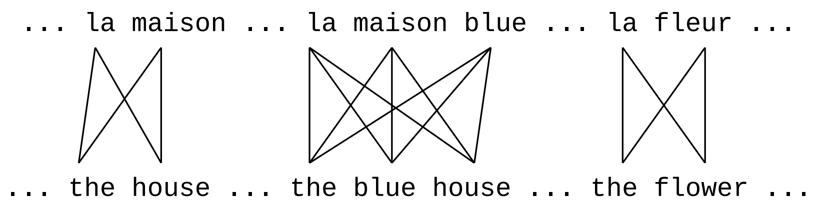
The EM algorithm usually works as follows:
1. The first step is model initialization. In our case, we give an initial value to each lexical translation parameter. Most of the time people would like to choose uniform distribution as initial values because at the beginning we assume every translation candidate is equally likely. But this will change as EM algorithm iterates over the data.
2. Expectation step: We apply the model to the data, to calculate the value of the latent variable, which is the most likely alignment a.
3. Maximization step: We learn the model from the data, i.e. re-estimate the lexical translation parameters.
4. Since this is an iterative method, we iterate steps 2 and 3 until it converges. In our case, convergence usually takes 5 to 10 iterations.

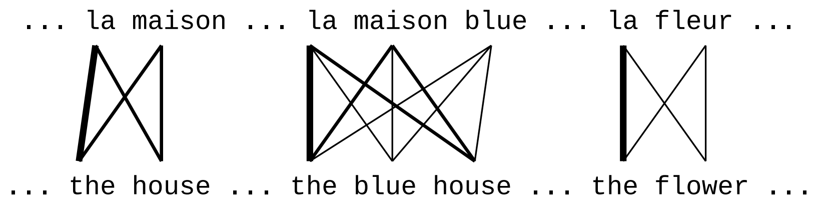
A visualization of the EM algorithm looks like Figure 1.4 (Philipp Koehn 2010), where all the alignments start to be equally likely but some of them gradually become more likely as the algorithm iterates through the data.

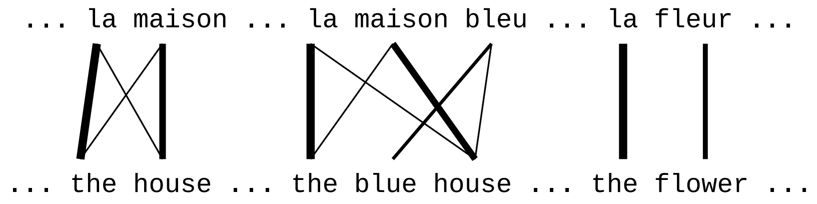
For the E step, we need to calculate $p(a|\mathbf{e}, f)$, this can be derived by the definition of conditional probability as follows:

$$p(a|\mathbf{e}, \mathbf{f}) = \frac{p(\mathbf{e}, a|\mathbf{f})}{p(\mathbf{e}|\mathbf{f})} \tag{1.10}$$

The numerator of equation 1.10 is already given before in the definition of IBM model 1, but we still need to derive the expression of the denominator, which is $p(\mathbf{e}|\mathbf{f})$. We can derive $p(\mathbf{e}|\mathbf{f})$ as follows:


... la maison ... la maison blue ... la fleur ...
... the house ... the blue house ... the flower ...
(a) EM Iteration 1


... la maison ... la maison blue ... la fleur ...
... the house ... the blue house ... the flower ...
(b) EM Iteration 2


... la maison ... la maison bleu ... la fleur ...
... the house ... the blue house ... the flower ...
(c) EM Iteration 3

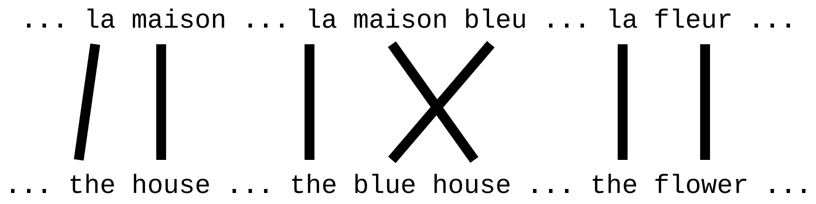

... la maison ... la maison bleu ... la fleur ...
... the house ... the blue house ... the flower ...
(d) EM Iteration 4

Figure 1.4: EM Algorithm, as demonstrated in SMT textbook (Philipp Koehn 2010)

$$\begin{aligned}
p(\mathbf{e}|\mathbf{f}) &= \sum_{a} p(\mathbf{e}, a|\mathbf{f}) \\
&= \sum_{a(1)=0}^{l_f} \dots \sum_{a(l_e)=0}^{l_f} p(\mathbf{e}, a|\mathbf{f}) \\
&= \sum_{a(1)=0}^{l_f} \dots \sum_{a(l_e)=0}^{l_f} \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)}) \\
&= \frac{\epsilon}{(l_f+1)^{l_e}} \sum_{a(1)=0}^{l_f} \dots \sum_{a(l_e)=0}^{l_f} \prod_{j=1}^{l_e} t(e_j|f_{a(j)}) \\
&= \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)
\end{aligned} \tag{1.11}$$

Let's now plug the equation 1.11 back into the equation 1.10, we can get the alignment probability as follows:

$$
\begin{aligned}
p(a|\mathbf{e}, f) &= \frac{p(\mathbf{e}, a|\mathbf{f})}{p(\mathbf{e}|\mathbf{f})} \\
&= \frac{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})}{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)} \\
&= \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=1}^{l_f} t(e_j|f_i)}
\end{aligned}
\tag{1.12}
$$

For the M step, we need to update the lexical translation probabilities, which is to collect different translation counts over all possible alignments, times their corresponding probability. Formally, this can be defined as follows:

$$
c(e|f; \mathbf{e}, \mathbf{f}) = \sum_a p(a|\mathbf{e}, \mathbf{f}) \sum_{j=1}^{l_e} \delta(e, e_j)\delta(f, f_{a(j)}) \tag{1.13}
$$

The delta function $\delta(x, y)$ equals to 1 if x = y and 0 otherwise. We can further plug in equation 1.12, which will give us the following expression:

$$
c(e|f; \mathbf{e}, \mathbf{f}) = \frac{t(e|f)}{\sum_{i=0}^{l_f} t(e|f_i)} \sum_{j=1}^{l_e} \delta(e, e_j) \sum_{i=0}^{l_f} \delta(f, f_i) \tag{1.14}
$$

After we get those counts, we can re-estimate the lexical translation probabilities by

$$
t(e|f; \mathbf{e}, \mathbf{f}) = \frac{\sum_{(\mathbf{e},\mathbf{f})} c(e|f; \mathbf{e}, \mathbf{f})}{\sum_e \sum_{(\mathbf{e},\mathbf{f})} c(e|f; \mathbf{e}, \mathbf{f})} \tag{1.15}
$$

Let's say after five iterations we are done with the training process, now how can we find the most likely alignment with a bunch of lexical translation probabilities at hand? Eventually the problem we were trying to solve is to find the alignment. For IBM Model 1, it is easy to show that the most likely alignment is given by:

$$
\hat{a} = argmax_a \prod_{j=1}^{l_e} t(e_j|f_{a(j)}) \tag{1.16}
$$

Above are the fundamentals of IBM model 1 and how to learn the parameters in our case. We will have more detailed explanation in later chapters.

Limitation

As the most basic model in the whole series of IBM models, the flaws of IBM model 1 are obvious, and we just list some of them here:

1. Only one-to-one and one-to-many type of alignments are modelled in IBM model 1. IBM model 1 assumes that for each word in the target sentence, there is at most one source word in the original sentence. But this is not true in real word since we often see a

phrase, which has several words, in the source sentence get translated into one word in the target sentence. This type of many-to-one alignment can not be captured by IBM model 1. And similarly, many-to-many alignment is not well modelled, either, which causes the IBM model 1 to be naturally flawed.

2. The position of the words in the target sentence are not considered. According to IBM model 1, two different target sentences generated with the same set of words are equally good although the word order can be pretty different. We can easily tell this is not true because in real world translation, word order is of great importance, which directly decides if the translation makes sense. The tendency we often observed that a contiguous phrase in the source language often gets translated into another contiguous phrase in the target language is not modelled.

3. The fertility of each input words is not modelled. For each input word in the source sentence, their ability to generate output words is different, which means for some words, they usually get translated into two words in the target language while for others, maybe one word is more frequently seen. Still, IBM model 1 does not provide any assumptions about the fertility problem.

Most of the limitations for IBM model 1 have been further addressed and fixed in higher IBM models. However, IBM model 1 provides us with a simple convex optimization procedure to learn word alignments which we can augment to learn a joint model of alignment and alignment types as we shall see later.

1.3.3 Generative Model and Discriminative Model

In the following chapters, we will propose two different models to find word alignments and predict the corresponding alignment types. And we will frequently refer to the term generative model and discriminative model. Thus, to make things clear, here we give a brief explanation to these two terms.

Generative Model: As the name suggests, generative model is a model that models the joint distribution of the observed data (the sentence pair) and the output labelling (the hidden alignment). For example, IBM model 1 is an example of generative model. In IBM model 1 we want to generate a target sentence from the source sentence. We achieve this by breaking the process into smaller steps, which models the lexical translation, and then combines the probability distributions for each aligned word pair to derive the probability of the whole sentence pair. A generative model is a complete probabilistic model for all variables.

Discriminative Model: In contrast with generative model which models the joint distribution of input and output random variables is what we call discriminative model which does not model the input random variables, but rather models the output variables conditional on the input. It is also called a conditional model because it is often used to model the dependency of an unobserved variable Y on an observed variable X. This is done by modelling the conditional probability distribution $p(Y|X)$, which can be used to predict Y given X.

To conclude, generative models are more flexible in terms of modelling the dependencies in observed data and are typically used in unsupervised learning while supervised learning tasks such as classification and regression, discriminative models perform better than generative models as they can use log linear or non-linear combinations of features in the input and they don't need to model the data distribution but rather they can directly minimize classification or regression loss on prediction of the output variables

1.4 Alignment Types

1.4.1 11 Different Alignment Types

Since alignment type is a new concept that requires some linguistic knowledge, we create a section here to give a more clear explanation of the 11 alignment types we use which are also used in the LDC Word Alignment and Tagging data. The 11 alignment types are shown in Table 1.3 below. And we provide some examples for each of the alignment types to give you a taste of their meanings:

Table 1.3: 11 different alignment types and their meanings

abbreviation	meaning
SEM	Semantic link
FUN	Function link
PDE	DE-possessive link
CDE	DE-clause link
MDE	DE-modifier link
GIF	Grammatically Inferred Function link
GIS	Grammatically Inferred Semantic link
COI	Context Inferred link
TIN	Translated Incorrect link
NTR	Not Translated link
MTA	Meta word link

Semantic links

This is the link between content words/phrases of source and translation, indicating direct equivalence. Content words are words that carry rich lexical meanings. For example, for English, content words are usually nouns, verbs, adjectives and adverbs. Different languages have different set of content words. For Chinese, there are more types of content words than those in English. However, we just simply adopt the English standard. If the words on both side of the alignment are content words, then the alignment type will be SEM. Some examples may look like:

中国 → China
特别 → special

Function links

This is the link between function words/phrases of source and translation, indicating function words are aligned between two languages. Function words are those that don't have domain specific lexical meaning but instead serve to represent the relationships with other words in the sentence. If the word on either side of the alignment is a function word, then the alignment type will be FUN. Note that punctuations are also covered in this type. Some examples may look like:

和 → and
等 → among other things
。 → .

DE-possessive links

Alignment types PDE, CDE and MDE are designed to handle the different features of the Chinese word 的. The Chinese word 的 in DE-possessive links assume the possession function in a modifying relationship. For example:

中国政法大学 **的** 教授 → Professor **at** Chinese University of Political and Law
彭德怀 **的** 高度警惕 → Great attention **from** Peng Dehuai

DE-clause links

Chinese word 的 in DE-clause links usually starts a modifying clause at the English side. For example:

对于经历过战争 **的** 人来说 → To those **who** have experienced war
可是让他们没想到 **的** 是 → but **what** they had not expected was

DE-modifier links

Chinese word 的 is frequently used in modifying relationship between words or constituents. This word can be translated differently and flexibly. When it does not assume the possessive or clause function during translation, it is the regarded as a DE-modifier. In this case, it is usually translated into prepositions in English. Some examples may look like:

数控技术 **的** 重要性 → The importance **of** numerical control technology

Grammatically Inferred Function links

Grouping or attaching function words to their head words can show word dependency relations. Links can still be of a semantic or function nature with extra words attached. The attached words can be functional or contextual, providing grammatical or contextual/semantic clues. The missing words on the other side of the link then can be inferred grammatically or contextually. Thus, according to how missing words are inferred, we have grammatically inferred links and contextually inferred links.

In grammatically inferred function links, stripping off extra words we get pure function links. Some examples may look like:

$$\text{的。} \rightarrow .$$
$$\text{一个} \rightarrow \text{a}$$

Grammatically Inferred Semantic links

In grammatically inferred semantic links, stripping off extra words results in pure semantic links. Some examples may look like:

$$\text{把... 机床} \rightarrow \text{machine tools}$$
$$\text{将... 成果} \rightarrow \text{success}$$

Where the Chinese words 把 and 将 don't actually contribute any lexical meaning but instead are attached for grammatical reasons, and stripping off them results in pure semantic links.

Contextually Inferred links

In contextually inferred links, the extra words attached to one side of the alignment are obligatory for the context. Without them, the grammatical structure might still be acceptable, but it is not semantically sensible. Such extra words are usually triggered by its co-occurring word or collocation word. The missed words on the other side of the link will be accordingly implied by word association or collocation. Some examples may look like:

$$\text{欢迎收看} \rightarrow \text{welcome}$$
$$\text{会见活动} \rightarrow \text{meeting}$$

In the examples above, when you say *Welcome to our show*, you actually imply *Welcome to watch our show*. Thus, an extra Chinese word 收看 which means *watch* is attached to another side of the alignment.

TIN, NTR and MTA

Those three link types are designed to handle the various errors occurred in the translation process, such as incorrectly translated, not translated and meta word. TIN stands for translated incorrectly link, and an example looks like:

$$\text{最好} \rightarrow \text{can}$$

In this pair of aligned words, the Chinese word actually means *would better* while the English word is *can*, which is not a very proper translation. However the Chinese word is still aligned to the English word because that's the best it can get in the English sentence. NTR stands for not translated link. A typical NTR would look like:

$$\text{此} \rightarrow$$

This usually happens when some words are dropped during the process of translating a Chinese sentence into an English sentence. This is not rare since the source Chinese sentence can have words with redundant meaning or the target English sentence just does not have an equivalent expression. MTA stands for meta words link. It was designed to handle some special characters in the corpus, such like & quot and ¬. They usually appear in the web context.

1.4.2 Dataset

Throughout our experiments, we used two different datasets:

The first one is the Word Alignment and Tagging corpus released by the LDC in 2014. It is called the GALE Chinese-English Word Alignment and Tagging corpus. Specifically, the releases we used in this series are:
1. LDC2012T16[17], GALE Chinese-English Word Alignment and Tagging Training Part 1 – Newswire and Web
2. LDC2012T20[18], GALE Chinese-English Word Alignment and Tagging Training Part 2 – Newswire
3. LDC2012T24[19], GALE Chinese-English Word Alignment and Tagging Training Part 3 – Web
4. LDC2013T05[21], GALE Chinese-English Word Alignment and Tagging Training Part 4 – Web
5. LDC2013T23[20], GALE Chinese-English Word Alignment and Tagging – Broadcast Training Part 1
6. LDC2014T25[22], GALE Chinese-English Word Alignment and Tagging – Broadcast Training Part 2

There are about 22k parallel sentences in the 5 releases altogether. We held out 2k sentences as the test set and used the remaining 20k as the training set. This dataset does not only include the parallel sentence aligned data, but also the gold alignments as well as the alignment types (tags) for those alignments.

The second dataset we used is the Hong Kong parliament proceedings (Hansards). We use 1 million sentences of the HK Hansards. However, in this dataset the gold alignments and tags are not provided. Only the sentence aligned Chinese and English text are given.

1.5 Contributions

The three main contributions of this thesis are:

1. We come up with a way to predict the alignment type given an aligned word pair and explore different feature types that can be used to improve the accuracy of prediction.

2. We introduce a new joint task which is to simultaneously predict word alignment and alignment types. This task has not been explored by anyone else before. We give two models, one generative and one discriminative, to solve this joint task, and show that the word alignments produced in this joint task are of higher accuracy than what was found in traditional word alignment task alone.

3. We show that such alignment type information can be used as an effective feature in generating word alignments and may be beneficial to the following machine translation process.

1.6 Thesis outline

The rest of this thesis will present the following contents:

Chapter 2 tries to solve a sub-task of the original joint problem we proposed, which is to predict the alignment type given a pair of aligned words.

Chapter 3 discusses two models we built to solve the original problem, one is generative and another is discriminative. Discussion goes all the way from the birth of idea, the definition and derivation of the formulas, down to the implementation and evaluation of the models

Chapter 4 is the final summary of the whole thesis. In this chapter, we repeat the key conclusions drawn from all the previous chapters and point out the possible future directions for continuing research.

Chapter 2

Alignment Type Prediction

Jointly computing the word alignment and alignment type is a harder problem than the intermediate problem of simply predicting the alignment type given a pair of aligned words. In computer science, a very useful strategy to tackle a hard problem is called divide and conquer, i.e. break down a large problem into smaller pieces, solve them one by one, and then combine the results. Similarly, here instead of directly solving the original problem, we first discuss the solution to a smaller sub-problem, which is how to predict the alignment type given a pair of aligned words.

2.1 Problem Definition

Formally, given an aligned word pair (e, f), what is the most likely type for this word alignment $e \to f$? Recall that we have 11 different kinds of alignment types in total. They form the following set {SEM, FUN, PDE, CDE, MDE, GIS, GIF, COI, TIN, NTR, MTA}

2.2 Logistic Regression

In real life, we often need to identify which category a given object belongs to and this type of problem is called classification. It is not hard to see our problem of alignment type prediction is actually also a problem of classification because all 11 types of alignments are given and the tag between a given word pair must fall into one of the 11 types.

Classification techniques are well developed in machine learning. And there are many different ways to solve a classification problem such as SVM, Nearest Neighbours, Random Forest, Logistic Regression, etc. However, we would like to use a method that can give us not only the alignment type label, but also the probability of being classified as this type. This is for the convenience of using this probability in later statistical models to solve the original problem. Thus, among those classification techniques, in order to get the full probability distribution, we decide to use Logistic Regression to solve our alignment type prediction task.

What is Logistic Regression?

Logistic Regression, contrary to what its name suggests, is a log linear model for classification rather than regression. A Logistic Regression model takes an input of any value, and maps it to a number between 0 and 1 and thus can be interpreted as a probability. This kind of mapping is done by a special function called logistic function, and it can transform any value from negative to positive infinity to some number between 0 and 1. The logistic function is defined as below:

$$\delta(t) = \frac{e^t}{e^t+1} = \frac{1}{1+e^{-t}} \tag{2.1}$$

A graph of the logistic function is shown in Figure 2.1

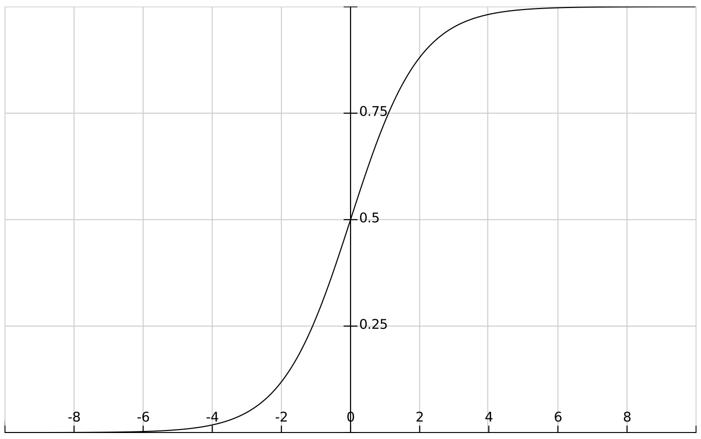


Figure 2.1: the graph for logistic function

t is a linear combination of our input variable x, t can be expressed as

$$t = \omega_0 + \omega x \tag{2.2}$$

We plug back the expression for t into the logistic function and now the logistic function can be written as:

$$F(x) = \frac{1}{1+e^{-(\omega_0+\omega x)}} \tag{2.3}$$

Given the input variable x we want to predict the label/category of x, denoted by Y. The value of $F(x)$ can be interpreted as the probability of x being labeled as 1. Thus if $F(x) >= 0.5$, we should predict $Y = 1$, otherwise $Y = 0$.

This also means to predict 1 whenever $\omega_0 + \omega x >= 0$ and 0 otherwise. Thus, the logistic function gives us a linear classifier. The solution of $\omega_0 + \omega x = 0$ is actually the decision boundary which divides the two different categories 0 and 1. And the shape of the decision boundary depends on the dimensionality of input variable x. If x is one dimensional, the decision boundary is a point; If x is two dimensional, the decision boundary is a line, etc. And the category probability depends on the distance from a certain point to the boundary.

Logistic Regression is one of the most commonly used techniques for discrete data analysis because it is simple but works surprisingly well. And it makes stronger and more detailed predictions due to the probabilities it can provide. Because of the probabilities, we can actually fit the logistic regression model in a different way than a normal classifier.

Model Fitting

Given the fact that Logistic Regression provides us probabilities, we can fit the model by using maximum likelihood. The first step is to express the likelihood function. Let's assume we have a set of training data points x_i and for each of them there is a category label y_i, that can be 0 or 1. The probability of a point x_i being labeled as 1 is $p(x_i)$ and $1 - p(x_i)$ for being labeled as 0. Thus, the total likelihood for the set of data points is a product over them:

$$L(\omega_0, \omega) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} \tag{2.4}$$

If we take the log of the above equation, we can get the log likelihood of the data:

$$\begin{aligned} l(\omega_0, \omega) &= \sum_{i=1}^{n} log(p(x_i)) + (1 - y_i)log(1 - p(x_i)) \\ &= \sum_{i=1}^{n} log(1 - p(x_i)) + \sum_{i=1}^{n} y_i log\frac{p(x_i)}{1 - p(x_i)} \\ &= \sum_{i=1}^{n} log(1 - p(x_i)) + \sum_{i=1}^{n} y_i(\omega_0 + x_i \cdot \omega) \\ &= \sum_{i=1}^{n} -log(1 + e^{\omega_0 + x_i \cdot \omega}) + \sum_{i=1}^{n} y_i(\omega_0 + x_i \cdot \omega) \end{aligned} \tag{2.5}$$

Now, we want to find the set of parameters that will maximize the likelihood of the data. Usually, to find the maximum likelihood estimates, we can take the derivative with respect to the parameters, then set the derivatives to zero and solve for the parameters. For example, let's take the derivative respect to one parameter, ω_j

$$\begin{aligned}\frac{\partial l}{\partial \omega_j} &= -\sum_{i=1}^{n} \frac{1}{1+e^{\omega_0+x_i\omega}} \cdot e^{\omega_0+x_i\omega} \cdot x_{ij} + \sum_{i=1}^{n} y_i \cdot x_{ij} \\ &= \sum_{i=1}^{n} (y_i - p(x_i;\omega_0,\omega))x_{ij}\end{aligned} \tag{2.6}$$

We can take the derivatives with respect to the other parameters to obtain the gradients for those parameters. However we cannot directly set the equations to zero and solve them because the equations are transcendental and no closed-form solution exists. However, we can still use some gradient methods to solve this type of optimization problem approximately, such as Newton's method or iterative re-weighted least squares (IRLS). In such methods we start with an initial guess for the parameters and iteratively improves it. We do not discuss the detail of the optimization algorithm (IRLS) here since it is not the focus of this thesis.

Logistic Regression for multiple categories

All the discussion above assumes that we only have two different class labels. But this is not true in our case. We have eleven different types of linguistic alignment types, which are eleven distinct output labels. In general we have two ways to solve the multi-class problem for logistic regression.

One is called One-Vs-Rest, which means for every class C_k, build a classifier between C_k and all the other classes. Then combine all the binary classifiers. For an input variable x, run it against every binary classifier and return the class that maximizes the log likelihood.

Another is the real Multiclass logistic regression, where we use a softmax function instead of logistic sigmoid and then learn the parameters ω towards the cross entropy loss. The softmax function looks like below:

$$p(y=k|x) = \frac{e^{\omega_k^T x}}{\sum_{i=1}^{K} e^{\omega_i^T x}} \tag{2.7}$$

Each class will have it own parameters and we learn a set of parameters $\{\omega_1, \omega_2, ...\omega_K\}$. When the total number of classes K equals to 2, the above equation reduces to the original logistic function in equation 2.3.

Implementation using Scikit-learn

Instead of building up a logistic regression classifier from scratch, the widely used Python based machine learning toolkit scikit-learn provides us with an efficient and easy-to-use implementation. The implementation of logistic regression in scikit-learn can be accessed from class LogisticRegression. This implementation can fit a multiclass (One-Vs-Rest) logistic regression with L2 regularization. It minimizes the following cost function:

$$min_{\omega,c}\frac{1}{2}\omega^T\omega + C\sum_{i=1}^{n} log(e^{-y_i(X_i^T\omega+c)}+1) \tag{2.8}$$

Scikit-learn's implementation uses liblinear solver which cannot learn a true multinomial (multiclass) model. Instead, the optimization problem is decomposed in a One-vs-Rest fashion so separate binary classifiers are trained for all classes. This happens under the hood, so LogisticRegression instances using this solver behave as multiclass classifiers. And the parameter C can be selected using the grid search function provided by scikit-learn.

2.3 Feature types

Now we have been armed with the basic weapon, logistic regression, to tackle the classification task. Let's think about what type of features $\mathbf{x}$ we can use to predict the alignment link type $\mathbf{y}$. Without further ado, one can come up with the most basic feature type, namely a pair of Chinese word and English word, (c_0, e_0). However, this may not be strong enough to give us high prediction accuracy. Besides considering the word pair itself, we can also take the context into consideration. This is to say, besides using the current word, one can also use another one or two words before or after it. An example would look like (c_{-1}, c_0, e_0) which also uses the previous one word on the Chinese side. In order to make the most use of the features and make sure the classifier will be able to generalize to unseen data, we can also consider some more transferable feature types like Part of Speech (POS) tags. Instead of using the word pair itself, we can also use the POS tags pair. Similarly, we can also use only the first five letters of the English word, which acts like a stem of the original word. This also helps to generalize the feature. Finally, we come up with a total of 22 types of different features, shown in the Table below:

We can aggregate any number of the feature types above, hopefully to achieve even better predication results.

2.4 Experiments and Results

We train the logistic regression classifier using the 20k LDC word alignment and tagging data. It contains Chinese-English parallel sentences, as well as the gold alignments for the sentence pairs and the linguistic tags for every word alignment. Some statistics about the training dataset are shown in the Table 2.3 below. We can see that among the 396,743 word alignments, the semantic link (SEM) makes up the largest number, followed by grammatically inferred functional link (GIF), function link (FUN), grammatically inferred semantic link (GIS), and other types of alignments.

Number	Feature Type	Example
1	c_0, e_0	一定, sure
2	c_{-1}, c_0, e_0	所以, 一定, sure
3	c_{-2}, c_{-1}, c_0, e_0	null, 所以, 一定, sure
4	c_0, c_1, c_2, e_0	一定, 要, 照顾, sure
5	c_0, e_-1, e_0	一定, sure, to
6	c_0, e_0, e_1	一定, sure, to
7	c_0, e_{t_0}, e_{t_1}	一定, JJ, TO
8	$c_0, [e_0]_5, e_{t_1}$	一定, sure, to
......	......	
15	$c_{t_0}, e_{t_0}, e_{t_1}$	AD, JJ, TO
......	......	
22	$c_{t_{-1}}, c_{t_0}, c_{t_1}, e_{t_0}$	AD, JJ, TO, JJ

Table 2.1: 22 different feature types and examples

Type of Alignment	Number
SEM	159,277
GIS	81,235
FUN	97,727
GIF	12,314
PDE	1,421
COI	3,256
CDE	1,608
TIN	1,116
MDE	4,615
NTR	34,090
MTA	84

Table 2.2: 396,743 aligned word pairs in 20k training data

In order to get the different types of features we came up with in previous section, we follow the rules below to process the data:
1. To get the word pairs, we extract the word pairs from the parallel sentences using the gold alignment provided.
2. To get the part of speech Tags, we first annotate the 20k data with the Stanford POS tagger (Toutanova et al. 2000), and then extract the POS tags.
3. We ignore the gold alignment if the Chinese side of the gold alignment is not contiguous, which means it can't form one Chinese word. This usually happens in the types of many-to-one and many-to-many word alignments.

Number	Feature Type	Example	Accuracy	10-fold CV
1	c_0, e_0	一定, sure	69.5	67.8
2	c_{-1}, c_0, e_0	所以, 一定, sure	69.6	67.8
3	c_{-2}, c_{-1}, c_0, e_0	null, 所以, 一定, sure	69.7	67.8
4	c_0, c_1, e_0	一定, 要, sure	69.4	67.6
......	......			
7	c_0, e_0, e_1	一定, sure, to	71.6	69.2
8	c_0, e_0, e_{t_0}	一定, sure, JJ	74.9	73.4
9	$c_0, [e_0]_5$	一定, sure	71.3	69.3
......	......			
15	c_{t_0}, e_{t_0}	AD, JJ	70.6	70.5
...	......	......	......	......
Combo.	1+2+...+15	......	82.0	77.7
Combo.	1+2+...+22	......	**87.5**	**81.5**

Table 2.3: Different feature types, combinations and the prediction accuracy (in %)

After we get all the different types of features from the training data, we train a logistic regression classifier using these features and their combinations. To evaluate the performance of our classifier, we report two numbers, one is to directly test on the same training data after training, another is to do a 10-fold cross validation on the training data. And the results are shown in the Table 2.3 below:

The experiments show that when we combine all the 22 types of features, our logistic regression classifier gives us the best predictions of an accuracy of 81.5%. We will use this trained classifier for all the following experiments and models.

The last evaluation task we did is to evaluate the above logistic regression classifier we trained with 22 different feature types on another 2k test data, which is held out from the training data. This represents how well the classifier works on unseen data. And the accuracy we got is **81.4%**, which is comparable to the result of 10-cross validation, indicating how well our classifier works for new data.

To give a more detailed analysis of the predicted labels, we come up with the confusion matrix in Table 2.4, where the vertical axis represents the actual classes and horizontal axis represents the predicted classes. In an ideal case, the numbers on the diagonal of the matrix should be as large as possible.

From the confusion matrix we can see that our logistic regression classifier works well for SEM, FUN, GIS, GIF, MDE, and CDE, since the numbers on diagonal are the largest in the row. However, for PDE, it seems hard for our classifier to distinguish it with MDE. And for COI and TIN, it's really easy to mis-predict them as other types.

Pre. / Act.	SEM	FUN	GIS	GIF	MDE	PDE	CDE	COI	TIN	NTR	MTA
SEM	14912	272	2896	28	0	0	0	18	0	0	0
FUN	292	8973	279	56	8	1	5	3	1	0	0
GIS	3306	222	12758	41	3	0	3	23	1	0	0
GIF	51	268	173	968	0	1	0	1	0	0	0
MDE	1	0	3	0	424	22	0	0	0	0	0
CDE	1	0	2	0	0	0	135	0	0	0	0
PDE	4	0	0	2	56	62	0	0	0	0	0
COI	238	8	293	1	0	0	0	28	0	0	0
TIN	73	18	60	8	0	0	0	0	10	0	0
NTR	0	0	0	0	0	0	0	0	0	0	0
MTA	0	0	0	0	0	0	0	0	0	0	0

Table 2.4: Confusion Matrix on test results, Actual Vs. Predicted

2.5 Summary

In this chapter, we described the problem of tag prediction given a pair of aligned words and gave a solution to this problem using logistic regression classifier. We explored 22 different types of features and their combinations for using a logistic regression classifier and evaluated them on our training and test data. We found out that using all the 22 feature types gives us the best result: 81.4% accuracy on unseen data and 81.5% accuracy using 10-fold cross validation on the training data..

Chapter 3

Joint Prediction of Word Alignment and Alignment Types

So far, we have already shown how to predict the alignment type between a given pair of aligned words. With this component in hand, we can start to consider solving the original joint task. In this chapter, we will introduce two different models, one generative and one discriminative, to solve the problem of simultaneously finding word alignments and predicting the corresponding alignment types. Some background knowledge will be provided wherever it is necessary.

3.1 Generative Model

3.1.1 Definition

Recall that in IBM Model 1, we break down the sentence translation probability into the product of all the individual word translation probabilities. The translation probability of a foreign sentence $\mathbf{f} = (f_1, ..., f_{l_f})$ of length l_f to an English sentence $\mathbf{e} = (e_1, ..., e_{l_e})$ of length l_e with an alignment function a is defined as follows:

$$p(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)}) \tag{3.1}$$

Our new generative model is an extension of IBM Model 1. We still obey the assumption that the sentence translation probability can be broken down as the product of word translation probabilities. However, in order to solve the joint problem, instead of modelling the pure sentence translation probability under a certain alignment function a, $p(\mathbf{e}, a|\mathbf{f})$, we take the alignment type probability into consideration and thus model the sentence translation probability with a certain alignment function a and a tagging function h, so that the alignment type for each alignment in a is predicted by h.

Thus, besides the alignment function $a : j \rightarrow i$, we introduce a new tagging function h and the value of $h(e_j, f_i)$ is the corresponding alignment type for alignment $j \rightarrow i$, i.e. one of the 11 linguistic tags. With the tagging function h, we can refine the translation probability of translating a foreign sentence $f = (f_1, ..., f_{l_f})$ of length l_f to an English sentence $e = (e_1, ..., e_{l_e})$ of length l_e with an alignment function $a : j \rightarrow i$ and tagging function $h : (e_j, f_i) \rightarrow h_k$, as follows:

$$p(\mathbf{e}, a, h|\mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e} \times N^{l_f+1}} \prod_{j=1}^{l_e} t(e_j, h(e_j, f_{a(j)})|f_{a(j)}) \tag{3.2}$$

And for each lexical translation probability, with the help of the chain rule, we can derive that:

$$t(e_j, h(e_j, f_{a(j)})|f_{a(j)}) = t(e_j|f_{a(j)}) \times t(h(e_j, f_{a(j)})|e_j, f_{a(j)}) \tag{3.3}$$

To make it simpler, we denote $f_{a(j)}$ as f_i and the value of $h(e_j, f_{a(j)})$ to be some tag h_k, so the formula now becomes:

$$t(e_j, h_k|f_i) = t(e_j|f_i) \times t(h_k|e_j, f_i) \tag{3.4}$$

This definition, like the pure IBM Model 1, is still based on the lexical translations. The formula above basically tells us that the probability of translating a source sentence to a target sentence with a certain alignment and a set of tags equals the product over the probability of translating each individual source word f_i into a target word e_j and the tag between this word pair is $h(e_j, f_i)$. The fraction before the product is for the purpose of normalization. There are $l_f + 1$ source words, including the special NULL word, and each of them can be aligned to any of the l_e target words. So totally there are $(l_f + 1)^{l_e}$ kinds of different alignments. And for each alignment there are N^{l_f+1} ways for tagging because there are N different type of tags, where N equals 11 in our context. The parameter ϵ is a constant to make $p(\mathbf{e}, a, h|\mathbf{f})$ a proper probability distribution, which means the sum over all alignments and tags should be 1.

To give you an example, we apply this new model to the example in Chapter 1 as shown in Figure 3.1, And it works as follows:

Five words are translated according to the alignments and tags above, so the overall probability should be the product over all the five lexical translation probabilities:

$$\begin{aligned} p(\mathbf{e}, a, h|\mathbf{f}) = & \frac{\epsilon}{5^4 \times N^5} \times t(\textit{Yesterday}, SEM|\text{昨天}) \\ & \times t(I, SEM|\text{我}) \times t(\textit{bought}, SEM|\text{买了}) \\ & \times t(a, FUN|\text{一本}) \times t(\textit{book}, SEM|\text{书}) \end{aligned} \tag{3.5}$$

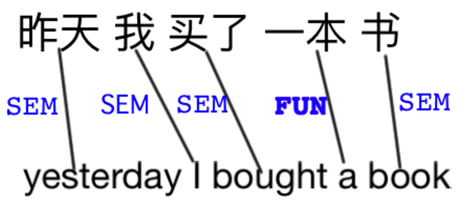


Figure 3.1: A Chinese sentence and its translation

3.1.2 EM for Generative Model

Similar to IBM Model 1, we still adopt the EM algorithm to learn the lexical translation probabilities. The derivation process follows closely the derivation of the EM for IBM Model 1. Recall that in the EM algorithm we typically do the following four steps:

1. Initialize the model
2. E step: apply model to the data
3. M step: learn new parameters from the data
4. Iterate Until convergence

In the Expectation Step, we need to compute the probability of a certain alignment with its set types given a sentence pair in the parallel text. That is to say, we want to compute $p(a, h|\mathbf{e}, \mathbf{f})$. Applying the chain rule gives us:

$$p(a, h|\mathbf{e}, \mathbf{f}) = p(a|\mathbf{e}, \mathbf{f}) \times p(h|a, \mathbf{e}, \mathbf{f}) \tag{3.6}$$

The $p(a|\mathbf{e}, \mathbf{f})$ part is easy to get since we already derived it in EM for IBM Model 1 (see Chapter 1). and we can just plug in the result value. It equals to:

$$\begin{aligned} p(a|\mathbf{e}, \mathbf{f}) &= \frac{p(\mathbf{e}, a|\mathbf{f})}{p(\mathbf{e}|\mathbf{f})} \\ &= \frac{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})}{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)} \\ &= \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=1}^{l_f} t(e_j|f_i)} \end{aligned} \tag{3.7}$$

As for the tagging probability part $p(h|a, \mathbf{e}, \mathbf{f})$, similarly, we break down the tagging probability over the whole sentence into individual words and take a product:

$$p(h|a, \mathbf{e}, \mathbf{f}) = \prod_{j=1}^{l_e} s(h_k|e_j, f_{a(j)}) \tag{3.8}$$

To distinguish from the translation probability t, here we use a different letter s to denote the tagging probability and h_k to denote the type of the word alignment $e_j, f_{a(j)}$. Thus, when we put the above two equations together, we get:

$$\begin{aligned} p(a, h|\mathbf{e}, \mathbf{f}) &= \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l_f} t(e_j|f_i)} \times \prod_{j=1}^{l_e} s(h_k|e_j, f_{a(j)}) \\ &= \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)}) \times s(h_k|e_j, f_{a(j)})}{\sum_{i=0}^{l_f} t(e_j|f_i)} \end{aligned} \tag{3.9}$$

As for the Maximization Step, we perform counting collection of all the lexical translations over various alignments as well as their tags. Those counts will be multiplied by their respective probabilities $p(a, h|\mathbf{e}, \mathbf{f})$, which we found in the above step. Formally, a count function c can be defined to reflect how we collect counts from a sentence pair $(\mathbf{e}, \mathbf{f})$ in parallel text that a certain foreign word f gets translated into an English word e with a tag h_k :

$$c(e, h_k|f; \mathbf{e}, \mathbf{f}) = \sum_{a,h} p(a, h|\mathbf{e}, \mathbf{f}) \sum_{j=1}^{l_e} \delta(e, e_j)\delta(f, f_{a(j)})\delta(h_k, h(e_j, f_{a(j)})) \tag{3.10}$$

Recall that the delta function $\delta(x, y)$ equals to 1 if x = y and 0 otherwise. We can further plug in equation 3.9, which will give us the following expression:

$$c(e, h_k|f; \mathbf{e}, \mathbf{f}) = \frac{t(e|f) \times s(h_k|e, f)}{\sum_{i=0}^{l_f} t(e|f_i)} \sum_{j=1}^{l_e} \delta(e, e_j) \sum_{i=0}^{l_f} \delta(f, f_i)\delta(h_k, h(e_j, f_i)) \tag{3.11}$$

The above count function c simply tells us: count the number of times that the source word f gets translated into English word e with alignment type h_k, and then multiply its probability.

Recall that in IBM Model 1 where no alignment type information is included, the count function is defined as:

$$c(e, |f; \mathbf{e}, \mathbf{f}) = \frac{t(e|f)}{\sum_{i=0}^{l_f} t(e|f_i)} \sum_{j=1}^{l_e} \delta(e, e_j) \sum_{i=0}^{l_f} \delta(f, f_i) \tag{3.12}$$

After collecting those counts, we can update the parameters, which are the lexical translation probabilities and the tagging probabilities. We have:

$$\begin{aligned} t(e, h_k|f) &= t(e|f) \times s(h_k|e, f) \\ &= \frac{\textit{number of times f translates into e}}{\textit{number of times f appears}} \times \frac{\textit{number of times f translates into e with tag } h_k}{\textit{number of times f translates into e}} \\ &= \frac{\textit{number of times f translates into e with tag } h_k}{\textit{number of times f appears}} \end{aligned} \tag{3.13}$$

Note the above formulas actually tell us that there are two ways to update $t(e, h_k|f)$, one is to directly use the final step, which is:

$$\begin{aligned} t(e, h_k|f) &= \frac{\textit{number of times f translates into e with tag } h_k}{\textit{number of times f appears}} \\ &= \frac{\sum_{(\mathbf{e},\mathbf{f})} c(e, h_k|f; \mathbf{e}, \mathbf{f})}{\sum_e \sum_{h_k} \sum_{(\mathbf{e},\mathbf{f})} c(e, h_k|f; \mathbf{e}, \mathbf{f})} \end{aligned} \tag{3.14}$$

Alternately, we can also use the product $t(e|f) \times s(h_k|e, f)$ to compute:

$$\begin{aligned} t(e, h_k|f) &= t(e|f) \times s(h_k|e, f) \\ &= \frac{\sum_{(\mathbf{e},\mathbf{f})} c(e, |f; \mathbf{e}, \mathbf{f})}{\sum_e \sum_{(\mathbf{e},\mathbf{f})} c(e, |f; \mathbf{e}, \mathbf{f})} \times \frac{\sum_{(\mathbf{e},\mathbf{f})} c(e, h_k|f; \mathbf{e}, \mathbf{f})}{\sum_{h_k} \sum_{(\mathbf{e},\mathbf{f})} c(e, h_k|f; \mathbf{e}, \mathbf{f})} \\ &= \frac{\sum_{(\mathbf{e},\mathbf{f})} c(e, |f; \mathbf{e}, \mathbf{f})}{\sum_e \sum_{(\mathbf{e},\mathbf{f})} c(e, |f; \mathbf{e}, \mathbf{f})} \times \frac{\sum_{(\mathbf{e},\mathbf{f})} c(e, h_k|f; \mathbf{e}, \mathbf{f})}{\sum_{(\mathbf{e},\mathbf{f})} c(e|f; \mathbf{e}, \mathbf{f})} \end{aligned} \tag{3.15}$$

In real world implementation, we can use either of the two ways as shown in equation 3.14 and 3.15, whichever is more convenient for processing the training data and collecting counts.

3.1.3 Pseudo Code

We provide the pseudo code for implementing the EM algorithm for our new generative model in real practice. There are some coding details and efficient tricks. For example, We store three count tables, one for the count of all foreign words f $count(f)$, one for the count of all possible word pairs (e,f) $count(e, f)$, and another for all possible word pair and their tags $count(e, f, h_k)$. the translation probabilities are actually not stored but instead computed on the fly every time we need it. And for model initialization, instead of giving every possible word pair an initial value, we initialize them only when we see them.

```
Input: Parallel sentence pairs (e,f) and linguistic tags h
Output: translation probability $t(e|f)$ and tag probability for an aligned word pair
        $s(h_k|e,f)$

Initialize $t(e|f)$ and $s(h_k|e,f)$ uniformly
while not converge do
    // initialize
    $count(e,f) = 0$ for all e,f
    $count(e,f,h) = 0$ for all e,f,h
    $count(f) = 0$ for all f
    for all sentence pairs and their tags (e,f,h) do
        // compute normalization
        for  each word e in e do
            $total(e) = 0$
            for each word f in f do
                $total(e) += t(e|f)$
            end
        end
        // collect counts
        for each word e in e do
            for each word f in f do
                $count(e,f) += \frac{t(e|f)}{total(e)}$
                $count(f) += \frac{t(e|f)}{total(e)}$
                if e and f are seen aligned with a tag $h_k$ then
                    $count(e,f,h_k) += \frac{t(e|f) \times s(h_k|e,f)}{total(e)}$
                end
            end
        end
    end
    // re-estimate probabilities
    for each foreign word f do
        for each English word e do
            $t(e|f) = \frac{count(e,f)}{count(f)}$
            for each tag type $h_k$ do
                $s(h_k,|e,f) = \frac{count(e,f,h)}{count(e,f)}$
            end
        end
    end
end
```

Figure 3.2: Pseudo Code for EM training

Figure 3.2 shows the pseudo code for the EM algorithm applied to our new generative model. We first uniformly initialize the probability distribution. Then we collect count according to equation 3.10 and 3.11. Eventually, we can re-estimate the probability distributions $t(e|f)$ and $s(h_k|e,f)$, whose product equals $t(e,h_k|f)$. In terms of time complexity and space complexity, the above algorithm is linear with the number of sentences and quadratic in sentence length, i.e. the number of words in sentence.

3.1.4 Finding the best alignment and tags

We obtain lots of word translation probabilities after the EM training is done. We can finally proceed to the last step of solving the original task of jointly predicting the best alignment with the best alignment type for each alignment. The optimal alignment-type combination is given by:

$$\hat{a}, \hat{h} = argmax_{a,h} \prod_{j=1}^{l_e} t(e_j, h|f_{a(j)}) \tag{3.16}$$

This equation tells us that we have to go through all the possible alignments and types to see which one yields the largest product over the lexical translation probabilities. Since each single lexical translation probability is independent of each other, we search through the possibilities that maximize the product. This is to say, for each source word f, we go through all the target words e and alignment types h_k to pick the one that makes $t(e,h_k|f)$ largest. Because this procedure is fairly simple, we omit the pseudo code here.

3.1.5 Remarks

Up till now, we have introduced our generative model for solving the joint task we defined. We started from the model definition, to how to train the model, and finally how to find the best alignment and tags using the trained model. In a word, this generative model is a modification of the original IBM Model 1. This kind of modification is natural because we have gold alignment and alignment types data at hand which makes it possible to collect the count of $count(e,f,h_k)$. And it makes sense to utilize the alignment type information because it will definitely add new evidence to the model.

3.2 Discriminative Model

3.2.1 Definition

Recall that in the generative model we just introduced, sentence translation probability is defined as:

$$p(\mathbf{e}, a, h|\mathbf{f}) = \frac{\epsilon}{(l_f+1)^{l_e} \times N^{l_f+1}} \prod_{j=1}^{l_e} t(e_j, h(e_j, f_{a(j)})|f_{a(j)}) \tag{3.17}$$

For each lexical translation probability, we have equation 3.18, where s is the tagging probability.

$$t(e_j, h_k|f_i) = t(e_j|f_i) \times s(h_k|e_j, f_i) \tag{3.18}$$

In the generative model, we use maximum likelihood estimation to model $s(h_k|e_j, f_i)$ and the value is updated during each EM iteration. But this is not the only way to do it. One can quickly relate to the logistic regression classifier we discussed before and it is a perfect source to provide us the tagging probabilities. And this is actually why we want to solve the sub-problem of alignment type prediction first.

Thus, in this discriminative model, we still follow the sentence translation model, the only difference is how we model the tagging probabilities. Instead of using maximum likelihood estimation, we use the pre-trained logistic regression model as a component to give us the tagging probabilities. The EM algorithm and all the equation derivations stay the same. But note that, in the EM algorithm for our discriminative model, we do not need to collect the counts for alignment types or update tagging probabilities anymore. Because this value now directly comes from the logistic regression classifier and will not change with the EM iterations.

3.2.2 Remarks

So far, we have proposed two models to tackle the problem of joint prediction of word alignment and alignment types.

- One is generative, which assumes sentence translation probability can be broken down into the product of individual word translation probabilities with types, and model the tagging probability with maximum likelihood estimation.
- Another is discriminative, which is based on the same assumption but instead use logistic regression to predict the alignment type probability.

We use the EM algorithm to learn the parameters, i.e. lexical translation probability t and alignment type probability s, filling the gap caused by incomplete data. Finally get the optimal word alignments and alignment types by maximizing the sentence translation probability.

3.3 Experiments and Results

3.3.1 Overview

To evaluate the performance of our two models, and compare with the baseline IBM model 1 that does not incorporate alignment type information, we designed the following experiments:

1. Train and test on the same 20k LDC data, with gold alignments and alignment types. This experiment is designed to test how well our new models work compare to baseline IBM model 1 when there is no OOV (out of vocabulary), i.e. all words have been seen previously.

2. Train on 20k LDC data, with gold alignments and alignment types, test on another 2k LDC data that are held out from the training data. This is to see how the models work when there are unseen data.

3. Train on 20k LDC data, with gold alignments and tags, and 1 million Hong Kong Hansards, without any additional annotation, test on another 2k LDC data that are held out from the training data. In order to better handle the OOV, we expand our training data to larger vocabularies. We first train the model on 20k LDC tagged data and then use it as the initialization to further train on the HK hansard.

4. To verify the significance level of our experimental results, we perform a significance test on the comparison of our model and the baseline model. The details will be described later.

Also, for each experiment, we measure the performance of our models and the baseline model on two different tasks:

1. One is the traditional word alignment task. We can compare our two models with the baseline model on this task to see if the alignment type information actually helps to find better word alignments.

2. Another is the new joint prediction task that we proposed, which is to predict both the word alignment and alignment types. This task is obviously harder because we need to get both of the word alignment and alignment type correct. Since the baseline model can't predict the alignment types, we can only do a comparison between our generative and discriminative models.

To evaluate the results on the above tasks, we calculate precision, recall and F-score (as defined in Chapter 1) by comparing the experimental results to the gold alignments and alignment types given in the 2k LDC test data.

3.3.2 Smoothing

When applying the models to test data, there must be some word pairs (c_0, e_0) and word pair with alignment types (c_0, e_0, h_k) that we have never seen in the training set. We call those OOV (out of vocabulary). However, we cannot simply return 0 for the unseen translation and alignment type probabilities because it will make the whole product 0. To solve the problem caused by OOV, there are two ways. One is to train the model using more data, hopefully to cover the unseen word pairs in the test data. Another is to use some smoothing techniques to smooth the model, avoiding the appearance of 0. Here we adopt the second approach, smoothing, because in real life we can never make sure that we have seen all the word pairs before.

To smooth the translation probability $t(e|f)$, we use the simple **Add-α Smoothing**. Original, the translation probability t equals to:

$$t(e|f) = \frac{c(e,f)}{c(f)} \tag{3.19}$$

It is easy to see that if the word pair (e, f) is never seen before, $c(e, f)$ will be zero. And if the word f never appears before, we cannot even do the division. To ensure both the numerator and denominator are not 0, we add a smoothing factor to them. In Add-α Smoothing, we do:

$$t(e|f) = \frac{c(e,f) + \alpha}{c(f) + \alpha \cdot V} \tag{3.20}$$

Where V is the vocabulary size, and α is a number smaller than 1. But what is a good value for α? We can determine this value by experimentation, i.e. by trying out different values and picking the one that gives us the best alignments on the held out data. Here we use the empirical value $\alpha = 0.005$.
To smooth the tagging probability $s(h_k|e, f)$, we use the **Back-off** strategy. Originally, the tagging probability is:

$$s(h_k|e,f) = \frac{c(e,f,h_k)}{c(e,f)} \tag{3.21}$$

Now besides using words, we back it off to part of speech (POS) tags:

$$s(h_k|e,f)^* = \begin{cases} \alpha \cdot \frac{c(e,f,h_k)}{c(e,f)} + (1-\alpha) \cdot \frac{c(t_e,t_f,h_k)}{c(t_e,t_f)} & \text{if } c(e,f) > 0 \text{ and } c(t_e,t_f) > 0 \\ s(h_k|t_e,t_f) & \text{if } c(t_e,t_f) > 0 \\ s(h_k|e,f) & \text{if } c(e,f) > 0 \\ 1e^{-13} & \text{otherwise} \end{cases} \tag{3.22}$$

Where t_e, t_f stands for the POS tag for English word e and Chinese word f. To get all those counts for POS tags, we need to train a similar model on pure POS tags. We do this by first labeling our parallel data with Stanford POS tagger and then train another model just on those POS tags.

Now we need to set a value for α. We would like to assign α and $1-\alpha$ to word based model and POS based model respectively, according to how reliable they are. It is obvious that we want to rely more on our word based model because it is more accurate and the POS based one is too general. Given the fact that $c(t_e,t_f)$ is much larger than $c(e,f)$, we set $\alpha = 1 - \frac{c(e,f)}{c(e,f)+c(t_e,t_f)}$, so that our word based model gets more weights.

If the word pair (e,f) was never seen before, then we use the second case in equation 3.22, which completely relies on the POS based model, and we compute it by:

$$s(h_k|t_e,t_f) = \begin{cases} \frac{c(t_e,t_f,h_k)}{c(t_e,t_f)} & \text{if } c(t_e,t_f,h_k) > 0 \\ P(h_k) & \text{otherwise} \end{cases} \tag{3.23}$$

Here $P(h_k)$ is the prior probability for alignment type h_k. It is calculated according to the Table 2.3 in Chapter 2.

$$P(h_k) = \begin{cases} 0.401 & \text{if } h_k = \text{SEM} \\ 0.246 & \text{if } h_k = \text{FUN} \\ 0.004 & \text{if } h_k = \text{PDE} \\ 0.004 & \text{if } h_k = \text{CDE} \\ 0.012 & \text{if } h_k = \text{MDE} \\ 0.205 & \text{if } h_k = \text{GIS} \\ 0.031 & \text{if } h_k = \text{GIF} \\ 0.008 & \text{if } h_k = \text{COI} \\ 0.003 & \text{if } h_k = \text{TIN} \\ 0.086 & \text{if } h_k = \text{NTR} \\ 0.002 & \text{if } h_k = \text{MTA} \end{cases} \tag{3.24}$$

Similarly, if the tag pair $c(t_e, t_f)$ was never seen before, then we use the third case in equation 3.22, which completely relies on word based models, and we compute it by:

$$s(h_k|e,f) = \begin{cases} \frac{c(e,f,h_k)}{c(e,f)} & \text{if } c(e,f,h_k) > 0 \\ P(h_k) & \text{otherwise} \end{cases} \tag{3.25}$$

For the last case, if neither (e, f) or (t_e, t_f) was seen before, which is very rare and unlikely, we just return a very small number $1e^{-13}$

Thus, for the experiment results we report below, the numbers are smoothed. Of course for experiment 1 in which we train and test on the same 20k data, we don't use any smoothing techniques since all the data are observed. But to better handle the experiments on the unseen test set, we adopt the two smoothing methods discussed above.

3.3.3 Experiments

Experiment 1: Train and test on the same 20K LDC data, with gold alignments and tags. As described in Chapter 1, we use F-score to evaluate the quality of our generated alignments compared with the gold alignments. A Higher F-score means better alignment quality. The results are shown in the table below:

		IBM 1	Discriminative	Generative	Gold
WA	Precision	27.7	28.1	39.4	38.6
	Recall	38.0	38.6	54.2	53.1
	F-score	**32.0**	**32.5**	**45.6**	**44.7**
WA+Tag	Precision		25.7	36.6	
	Recall		35.4	50.4	
	F-score		**29.8**	**42.4**	

Table 3.1: Train and test on the same 20k tagged LDC data, in %

In the table above, WA stands for the task of Word Alignment and WA+Tag stands for the joint task. And please note that for baseline IBM model 1, the output will be pure word alignments but for our generative and discriminative model, the output will be both word alignments and their types. Thus, for task WA, we only take the word alignment part and ignore their types, for task WA+Tag, we use them both.

In terms of finding the optimal word alignments, it's not hard to see from the above table that both of our generative and discriminative models perform better than the baseline IBM model 1 that does not incorporate the alignment types. It shows that the alignment type information actually helps to find better alignment.

As for the joint prediction task, our generative model does better than the discriminative model. This is understandable because in this experiment the test data is the same as the training data and the generative model memorizes all the word pairs and the alignment types it met before thus tend to give better predictions. but the difference is actually not that huge, if we only take a look at the alignment type prediction task, as shown in Table 3.2

Model	Num. of correct WA	Num. of correct WA+Tag	Accuracy%
Generative	20386	18951	93.0
Discriminative	14532	13331	91.7

Table 3.2: Alignment type prediction accuracy

Although these two accuracy numbers cannot be directly compared because the numbers of correct word alignment retrieved are different, we can still get a rough idea on how well the alignment prediction works.

If you notice there is a column called Gold in Table 3.1. This is the model built directly from the given gold alignment, which means during the EM algorithm we collect counts according to the gold alignment. The performance of this gold model is very competitive, but still lower than our generative model. This means the training on alignment types is not only injecting information about word alignments alone into our training procedure. It is providing another layer of information to our models.

Experiment 2: Train on 20K LDC data, test on another 2k held-out LDC data.
To give a more objective and fair evaluation to our models, we test the models trained in experiment 1 on another 2k unseen data. The smoothing techniques described before are used to handle the OOV. The results are shown in Table 3.3:

		Baseline	Generative	Discriminative
WA	Precision	31.9	32.8	31.1
	Recall	43.6	44.9	42.6
	F-score	**36.9**	**37.9**	**36.0**
WA+Tag	Precision		29.2	27.7
	Recall		39.9	37.9
	F-score		**33.7**	**32.0**

Table 3.3: Train on 20k, test on 2k, in %

For the word alignment task (WA), this time although our generative model is not dramatically better than our baseline model, it is still considerably better than the baseline, which is consistent with the previous experiment. However, this time our discriminative model performs worse than the baseline. This is understandable because in the discriminative model, the alignment type probabilities are actually not involved in the EM training process but instead are fixed values provided by the pre-trained logistic regression classifier.

Thus, we would expect the word alignment found by the discriminative model no better than those found by the pure baseline model when the accuracy of logistic regression is not extremely high.

Experiment 3: Train on 20k LDC data, with gold alignments and tags, and 1 million Hong Kong Hansards, without any additional annotation, test on another 2k held-out LDC data.

Note that in this experiment, we are trying to enhance the model performance on unseen data, getting better results than experiment 2. Thus, besides the 20k LDC data, we use another one million sentences from Hong Kong Hansards. Although the Hong Kong Hansards is not annotated with any gold alignment or alignment types, it can still augment our vocabulary. We call the model trained on 20k LDC data LDC model. And then use it as the initialization for another larger model trained on HK Hansards. We call this larger model augmented model.

However, when computing the translation probabilities $t(e|f)$, this augmented model will be negatively affected by the use of data in a different domain from the 20k training data we used to train the alignment type models. Also, while HK Hansards are a different domain that contains 1M sentence pairs and is much larger than 20k training set for alignment types (LDC data is more about news, web and broadcasting). Thus, in our first experiment, which directly uses the augmented word translation model, actually gives us slightly worse results compared to experiment 2. So we need a proper way to use the augmented model. Since the 2k test data is also from the LDC data, which is more similar to the LDC model, we would like to compute $t(e|f)$ using the LDC model first if the word pair (e, f) appears in the LDC model, otherwise we will use the augmented model to compute $t(e|f)$. This backoff strategy can be represented as:

$$t(e|f) = \begin{cases} \frac{c(e,f)}{c(f)} & \text{if } c(e,f) > 0 \\ \frac{c_A(e,f)}{c_A(f)} & \text{otherwise} \end{cases} \tag{3.26}$$

Here C_A represents the counts collected by the augmented model. It works just as a backoff option for the smaller LDC model. And using this backoff strategy we observed obvious improvements when test on the 2K unseen data, as shown in Table 3.4:

For the Word Alignment task (WA), this time our generative model has a much higher F-score than the baseline IBM model 1. Our discriminative model appears to be slightly worse than the baseline model. But all of the three models perform better than using only the LDC model as in experiment 2.

For the joint prediction task (WA+tag), both of our generative and discriminative models continue to be better than the baseline and gained better results than experiment 2. Our

		Baseline	Generative	Discriminative
WA	Precision	32.2	36.2	31.7
	Recall	44.0	49.5	43.4
	F-score	**37.2**	**41.8**	**36.7**
WA+Tag	Precision		32.1	28.1
	Recall		43.9	38.4
	F-score		**37.1**	**32.5**

Table 3.4: Performance on 2k unseen data, in %, backoff using augmented model

generative model performs better than the discriminative model because generative model gets more correct alignments and the test data are in the same domain as the training set so that the discriminative classifier does not really outperform the generative one .

If we do a detailed analysis on separate alignment types, we can get the prediction precision, recall and F-score for each alignment type as shown in Table 3.5:

		Discriminative			Generative		
Type	%	Precision	Recall	F-score	Precision	Recall	F-score
SEM	40.5	42.0	53.6	47.1	37.9	60.7	46.7
FUN	24.9	30.0	56.9	39.3	47	62.9	53.8
GIS	20	2.2	3.1	2.6	3.3	5.1	4.0
GIF	3	29.5	35.6	32.3	52.3	53.3	52.7
COI	0.8	5.9	1.0	1.7	11.7	4.6	6.6
PDE	0.3	16.7	37.9	23.2	30.3	35.5	32.7
CDE	0.4	21.3	30.9	25.2	28.5	66.2	39.8
MDE	1.2	14.4	54.1	22.8	23.7	59.4	33.9
TIN	0.3	5.9	1.0	1.6	5	1.8	2.6
MTA	0.01	0	0	0	0	0	0
NTR	8.6	0	0	0	0	0	0

Table 3.5: Precision, Recall and F-score on each alignment type

The second column is the percentage of each alignment type out of the whole test data. However, the above calculation in Table 3.5 is actually not a fair evaluation of the alignment type prediction task because the results are based on all the alignments produced, including the wrong ones. For example, for alignment type SEM, We got 15520 alignments labeled as SEM in the gold alignment, 28838 in the produced alignment, and 8972 in their intersection. Using these numbers we can get the precision, recall and F-score in Table 3.5. However, this seems a little bit problematic since the 28838 also contains wrong alignments, which means the alignment is already wrong, so there's no way to get the label right (no way to become the intersection part), some errors actually may not be the fault of the type prediction. We are underestimating the accuracy of alignment type prediction in this way.

So, what if we redo the computation based on the correct alignments only? if using only the correct alignments, we got the results in Table 3.6.

		Discriminative			Generative		
Type	%	Precision	Recall	F-score	Precision	Recall	F-score
SEM	40.5	95.5	53.6	68.7	95.6	60.7	74.2
FUN	24.9	98.4	56.9	72.1	98.5	62.9	76.8
GIS	20	15.2	3.1	5.2	20.6	5.1	8.2
GIF	3	90.5	35.6	51.1	95.9	53.3	68.5
COI	0.8	23.1	1.0	1.9	48.3	4.6	8.4
PDE	0.3	75.8	37.9	50.5	67.7	35.5	46.6
CDE	0.4	100	30.9	47.3	96.8	66.2	78.6
MDE	1.2	87.5	54.1	66.8	87.3	59.4	70.7
TIN	0.3	50	1.0	1.8	33.3	1.8	3.4
MTA	0.01	0	0	0	0	0	0
NTR	8.6	0	0	0	0	0	0

Table 3.6: Per type results based only on correct alignments

This time, the precision and F-score seem to be boosted a lot. But this is also problematic since some incorrect alignments may be produced due to their wrong type predictions. We are overestimating the accuracy in this way.

Although we don't have a way to report exactly how well we did in alignment type prediction alone since there is a correlation between the alignment and alignment types found in the joint task. We can still see from the above two tables that we're doing well with alignment type SEM, FUN, GIF, PDE, CDE and MDE. And we have serious problem in recognizing alignment type GIS, COI and TIN. For MTA and NTR, we got none of them right either because there are too little data of this type or the alignment type itself is meaningless. We can consider removing these two types of labels. By observing this analysis, we will design new features specific to the alignment types that our current models are weak to capture and further improve the model performance.

Experiment 4: Significance test. Our generative model consistently works better than the baseline model in all the previous experiments above so we want to do a significance test between the generative model and the baseline model to make sure the improvements are stable and reliable. We did a significance test for the previous experiment 3.

For many evaluation metrics in NLP, assessing the significance is non-trivial since standard tests usually make assumptions like data should follow a certain type of distribution, but those assumptions are hard to meet in real experimental settings. To overcome this, we use the assumption-free approximate randomization framework (Yeh 2000, Noreen 1989).

The main idea is that if that the difference in performance between two sets of predictions is significant, random shuffling of the predictions will only very infrequently result in a larger performance difference. The relative frequency of this actually happening can be interpreted as the significance level of the difference.

Thus, we divide our 2k test set into 10 random sample, and run our generative model and the baseline model on those 10 samples. This gives us two sets of predictions and each set contains 10 observations. We feed those observations into the Sigf software and it reports the significance level. The 10 observations are shown in Table 3.7 below:

Observ.	tp1	tp2	tp+fp	tp+fn
1	1705	1927	5190	3839
2	1641	1844	5180	3806
3	1724	1928	5387	3938
4	1751	1976	5509	3977
5	1790	2000	5680	4117
6	1736	1943	5346	3922
7	1698	1921	5186	3773
8	1743	1968	5477	3990
9	1746	1975	5502	3969
10	1344	1525	4020	3046

Table 3.7: tp=true positive, fp=false positive, fn=false negative

tp1 stands for the true positive got from the baseline model and tp2 is for our generative model. tp+fp is actually the number of all generated alignments and tp+fn is actually the total number of all gold alignments. So those two numbers are the same for both models. And the 2-tailed p-value we got from the Sigf software, which uses 10 observations and 10,000 iterations, is **0.003**. Because 0.003 is much lower than the empirical significance level 0.05. So we can say that our improvements are significant at a significance level of 95%.

3.3.4 Remarks

In this section, we evaluate our two models from different perspectives.

Experiment 1 shows that our models using alignment types perform much better than the baseline when test data haven been seen before. And alignment type information helps to find better word alignments.
Experiment 2 and 3 shows the case of how unseen data are handled. In both of the experiments, our generative model using alignment type information still consistently shows superiority over the baseline model. And both of our generative and discriminative models work well in terms of predicting the alignment types.
Experiment 4 tells us that compared to the baseline model, the improvements we gained in our generative model are significant, consistent and reliable.

Chapter 4

Conclusion and Future Work

In this work, we use 11 different alignment link types based on annotated data released by the Linguistics Data Consortium to enrich word alignment. We proposed a new joint task of simultaneously predicting word alignment and alignment types. In the first part of this work, we focused on the sub-task of alignment type prediction given a pair of aligned words. We explored different type of features that can be used to do the prediction and trained a logistic regression classifier with 22 different types of features which is able to recover the alignment types with an accuracy of 81.4%.

In the second part of this work, we proposed two models, one generative and another discriminative, to solve the original joint prediction task. We break down the sentence translation probability to the product of lexical translation probabilities and we learn the model parameters using expectation maximization algorithm. For the generative model, we model the alignment type probability using maximum likelihood estimation and for the discriminative model, we directly use the logistic regression classifier we built before.

We compare our two models with a baseline model that is unaware of the alignment types in different experimental settings and results clearly tell us that the alignment type information helps to find better word alignments.

4.1 Future Work

As we mentioned before, this is actually the first time one uses typed alignment information to find better word alignment and do the joint prediction. This purpose of this work is just to verify the usefulness of the alignment types. Thus, there are many avenues for extension of this work.

First, the baseline translation model we used is the most basic one, IBM model 1. It is actually not that strong and is naturally unable to capture certain types of alignments. This leads to the relatively lower performance of all our models. Thus, one of the major work we can do in the future is to replace the baseline model with more advanced models and build our joint models on that.

Second, in terms of smoothing, the techniques we adopted are also simple. There are other more complicated and reasonable smoothing techniques in NLP that can be used here. And we believe those better smoothing techniques will further improve the performance of our models.

Third, to better handle the OOV, we use more data (1 million HK hansard) to train and hope that the knowledge we learned on the 20k LDC data can transfer to the larger data set. The larger dataset definitely helps to inject more vocabularies but is weak in terms of providing alignment type information because it is unlabeled. We can do transductive learning here to fill the gap and provide better prediction for alignment types.

Fourth, we've done some analysis about the model performance on each alignment types in the previous chapter. We will design new features specific to the alignment types that our current models are weak to capture and further improve the model performance.

Last but not the least, alignment type information should also be helpful to other NLP tasks such as projection of part of speech tags and dependency trees from a resource-rich language to a resource-poor language. We would expect typed word alignments to pay off in many NLP applications.

4.2 Contributions

The main contributions of this thesis work are as follows:

1. We came up with a new task of joint prediction of word alignment and alignment types. This is the first time people use alignment type information to help find better alignment.
2. We provided a method to predict alignment types given a pair of aligned words, explored different types of features that can be used to do the prediction.
3. We proposed two new models, one generative another discriminative, to solve the joint prediction task. And showed that the alignment found in our joint framework is better than those found in traditional word alignment task alone.
4. We see the hope for benefitting other NLP tasks using the alignment type information.

In conclusion, our joint model improves the word alignment quality and produces alignment types at the same time. We expect typed word alignments to benefit not only SMT, but also other NLP tasks that rely on word alignments.

Bibliography

[1] Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 582–590, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[2] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *COMPUTATIONAL LINGUISTICS*, 19:263–311, 1993.

[3] Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. Optimizing chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 224–232, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[4] David Chiang, Steve DeNeefe, and Michael Pust. Two easy improvements to lexical weighting. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 455–460, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[5] David Chiang, Kevin Knight, and Wei Wang. 11,001 new features for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 218–226, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[6] Tagyoung Chung and Daniel Gildea. Unsupervised tokenization for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 718–726, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[8] John Denero. Tailoring word alignments to syntactic machine translation. In *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-2007*, pages 17–24, 2007.

[9] George Foster, Roland Kuhn, and Howard Johnson. Phrasetable smoothing for statistical machine translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 53–61, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[10] Alexander Fraser and Daniel Marcu. Measuring word alignment quality for statistical machine translation. *Comput. Linguist.*, 33(3):293–303, September 2007.

[11] Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. Better word alignments with supervised itg models. 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: Volume 2 - Volume 2*, ACL '09, pages 923–931, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[12] Abraham Ittycheriah and Salim Roukos. A maximum entropy word aligner for arabic-english machine translation. In *In Proceedings of HLT-EMNLP*, pages 89–96, 2005.

[13] Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[14] X. Li, N. Ge, and S.M. Strassel. *Guidelines for Chinese-English Word Alignment*, 2009.

[15] X. Li, N. Ge, and S.M. Strassel. *Tagging Guidelines For Chinese-English Word Alignment*, 2009.

[16] Xuansong Li, Niyu Ge, Stephen Grimes, Stephanie Strassel, and Kazuaki Maeda. Enriching word alignment with linguistic tags. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*, 2010.

[17] Xuansong Li, Stephen Grimes, and Stephanie Strassel. *GALE Chinese-English Word Alignment and Tagging Training Part 1 – Newswire and Web*. Philadelphia: Linguistic Data Consortium, LDC2012T16. Web download file, 2012.

[18] Xuansong Li, Stephen Grimes, and Stephanie Strassel. *GALE Chinese-English Word Alignment and Tagging Training Part 2 – Newswire*. Philadelphia: Linguistic Data Consortium, LDC2012T20. Web download file, 2012.

[19] Xuansong Li, Stephen Grimes, and Stephanie Strassel. *GALE Chinese-English Word Alignment and Tagging Training Part 3 – Web*. Philadelphia: Linguistic Data Consortium, LDC2012T24. Web download file, 2012.

[20] Xuansong Li, Stephen Grimes, and Stephanie Strassel. *GALE Chinese-English Word Alignment and Tagging – Broadcast Training Part 1*. Philadelphia: Linguistic Data Consortium, LDC2013T23. Web download file, 2013.

[21] Xuansong Li, Stephen Grimes, and Stephanie Strassel. *GALE Chinese-English Word Alignment and Tagging Training Part 4 – Web*. Philadelphia: Linguistic Data Consortium, LDC2013T05. Web download file, 2013.

[22] Xuansong Li, Stephen Grimes, and Stephanie Strassel. *GALE Chinese-English Word Alignment and Tagging – Broadcast Training Part 2*. Philadelphia: Linguistic Data Consortium, LDC2014T25. Web download file, 2014.

[23] Percy Liang and Dan Klein. Online em for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 611–619, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[24] Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 104–111, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[25] I. Scott MacKenzie. How to report an f-statistic. `http://www.yorku.ca/mack/RN-HowToReportAnFStatistic.html`. Accessed: 2015-09-30.

[26] Robert C. Moore. Fast and accurate sentence alignment of bilingual corpora. In *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas on Machine Translation: From Research to Real Users*, AMTA '02, pages 135–144, London, UK, UK, 2002. Springer-Verlag.

[27] Robert C. Moore. Improving ibm word-alignment model 1. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[28] Radford M. Neal and Geoffrey E. Hinton. Learning in graphical models. chapter A View of the EM Algorithm That Justifies Incremental, Sparse, and Other Variants, pages 355–368. MIT Press, Cambridge, MA, USA, 1999.

[29] Eric W. Noreen. *Computer-intensive methods for testing hypotheses.* Wiley, New York, NY, USA, 1st edition, 1989.

[30] Franz Josef Och and Hermann Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 440–447, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[31] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51, March 2003.

[32] Sebastian Padó. *User's guide to `sigf`: Significance testing by approximate randomisation*, 2006.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[34] Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. Optimization with em and expectation-conjugate-gradient. pages 672–679, 2003.

[35] Cosma Shalizi. Lecture notes in logistic regression, April 2012.

[36] Artem Sokolov. Lecture notes in statistical machine translation, April 2015.

[37] Ben Taskar, Simon Lacoste-Julien, and Dan Klein. A discriminative matching approach to word alignment. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 73–80, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[38] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[39] Kristina Toutanova and Christopher D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, EMNLP '00, pages 63–70, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[40] Huihsin Tseng. A conditional random field word segmenter. In *In Fourth SIGHAN Workshop on Chinese Language Processing*, 2005.

[41] Yale University. Tests of significance. `http://www.stat.yale.edu/Courses/1997-98/101/sigtest.htm`. Accessed: 2015-09-30.

[42] Stephan Vogel, Hermann Ney, and Christoph Tillmann. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 836–841, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

[43] Wikipedia. Discriminative model, 2015. [Online; accessed 12 September 2015].

[44] Wikipedia. Expectation–maximization algorithm, 2015. [Online; accessed 21 October 2015].

[45] Wikipedia. Generative model, 2015. [Online; accessed 12 September 2015].

[46] Wikipedia. Logistic regression, 2015. [Online; accessed 21 October 2015].

[47] Wikipedia. Precision and recall, 2015. [Online; accessed 12 September 2015].

[48] Jinxi Xu and Jinying Chen. How much can we gain from supervised word alignment? In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 165–169, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[49] Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. Building a large-scale annotated chinese corpus. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–8, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[50] Alexander Yeh. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 947–953, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[51] Hui Zhang and David Chiang. Kneser-ney smoothing on expected counts. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 765–774, 2014.