Word Alignment and Smoothing Methods in Statistical Machine Translation: Noise, Prior Knowledge, and Overfitting

Tsuyoshi Okita

A Dissertation submitted in fulfilment of the requirements for the award of

Doctor of Philosophy (Ph.D.)

to

Dublin City University

School of Computing

Supervisor: Prof. Andy Way

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:

(Candidate) ID No.:

Date:

Contents

Abstract

This thesis discusses how to incorporate linguistic knowledge into an SMT system. Although one important category of linguistic knowledge is that obtained by a constituent / dependency parser, a POS / super tagger, and a morphological analyser, linguistic knowledge here includes larger domains than this: Multi-Word Expressions, Out-Of-Vocabulary words, paraphrases, lexical semantics (or non-literal translations), named-entities, coreferences, and transliterations. The first discussion is about word alignment where we propose a MWE-sensitive word aligner. The second discussion is about the smoothing methods for a language model and a translation model where we propose a hierarchical Pitman-Yor process-based smoothing method. The common grounds for these discussion are the examination of three exceptional cases from real-world data: the presence of noise, the availability of prior knowledge, and the problem of underfitting. Notable characteristics of this design are the careful usage of (Bayesian) priors in order that it can capture both frequent and linguistically important phenomena. This can be considered to provide one example to solve the problems of statistical models which often aim to learn from frequent examples only, and often overlook less frequent but linguistically important phenomena.

List of Tables

Chapter 1

Introduction

Machine Translation (MT) is the study of automatic translation that incorporates knowledge of both linguistics and statistics. As in other areas of Artificial Intelligence,[1] the influx of statistical approaches in the 1990's had a dramatic effect on MT, so much so that to this day, the main models are statistical. The study of such statistical methods applied to MT is known as Statistical Machine Translation (SMT). A deeper examination of applying Machine Learning methods may lead to further improvements in the quality of MT output. The research presented in this thesis attempts this application, and we examine one particular module *word alignment* and one particular method *smoothing* which applies to both the language model and the translation model.

Despite being rarely discussed in the MT community, the effort required goes beyond the most complex state-of-the-art Machine Learning algorithms. MT requires the conversion of a sequence of words in the source language into another sequence of words in the target language where 1) the length of the source-language sequence and that of the target-language sequence may be different,[2] and 2) the correspondences between elements in the source-language sequence and those in the

[1] Computer Vision is one such area where statistical approaches are most intensively used (Forsyth and Ponce, 2003); conversely, Computer Vision contributed to the progress of Machine Learning.

[2] Structured prediction algorithms handle input and output whose lengths are the same and whose correspondences are already assigned from the beginning. For example in a DNA sequence in biology, the input sequence and output sequence, which constitutes either A (Adenine), T (Thymine), G (Guanine), and C (Cytosine), are the same length and their correspondences match with their index.

EN	on this **particular** building
FR	dans ce bâtiment
EN	the health and safety legislation that it **actually** passes
FR	la reglementation en matière de santè et de securite qu' il vote.
EN	why are there **no fire instructions** ?
FR	comment se fait-il qu' **il n' y ait pas de consignes en cas d' incendie** ?
EN	there are two finnish channels and one portuguese **one**.
FR	il y a bien deux chaînes finnoises et une chaîne portugaise.

Table 1.1: Four simple examples that may break the *pair assumption*.

target-language sequence may be reordered. One fairly strong assumption made in SMT is that the translation model $P(\bar{e}|\bar{f})$ has always accomodated a pair of $\bar{e}$ and $\bar{f}$, which never lacks one side. This assumption radically reduces the complexity of the problem although this may yield some other problems. This thesis examines the methods within this assumption, untouched in other complex cases since this is really a difficult ultimate goal of SMT.

Note that we may say that Machine Learning perspectives are not identical with SMT perspectives, but these two may complement each other in the following sense. On the one hand, various particular MT technologies are developed for SMT such as phrase extraction, stack-based decoding, Hierarchical Phrase-Based Machine Translation (HPB-SMT), Minimum Error Rate Training (MERT), and so forth. On the other hand, the Machine Learning point of view concerns noise, prior knowledge, overfitting, statistical assumptions, and so forth. This thesis focuses on noise, prior knowledge, and overfitting.

Among others, one idea that radically reduces the overall computational complexity in SMT is that it assumes that an observed word / phrase always forms a pair. (We call this the *pair assumption* in this thesis.) A t-table and a translation model always accommodate source and target words / phrases. SMT is constructed based on this assumption. For example, any pair in the translation model is never lacking either side of the word / phrase. Under this assumption, we can write a pair of words / phrases $e_i|f_i$ using just one variable x. In the EM algorithm (Dempster et al., 1977), the latent variable $A_{i \to j}$ can be written in terms of this x. In the HMM algorithm (Baum et al., 1970;

Baker, 1975), the observation y can be written in terms of x as well. There is a slight difference in the notation between the SMT and Machine Learning literature, but if we convert $e_i|f_i$ to x, or x to $e_i|f_i$, this becomes transparent.

The starting point of this thesis is a close look at the word alignment component, that is the IBM Models of 1993 (Brown et al., 1993) and HMM Model of 1996 (Vogel et al., 1996). The first motivation comes from the Machine Learning level in the usage of EM and HMM algorithms which are already twenty or thirty years old (EM of 1976 and HMM of 1989). Although these algorithms are still popular given their simplicity, it is known today that they have some problems in terms of, for example, overfitting, noise (or outliers), prior knowledge, and extensibility. Nowadays, we know of various improved algorithms. Increasingly, Machine Learning algorithms are often designed for synthetic data where often do not go beyond the researcher laboratory. SMT has to handle real-life data. This may be a very good reason to look at the robustness of Machine Learning algorithms. One recommendation we describe in this thesis is the use of graphical models in terms of extensibility where we employ the MAP assignment framework for prior knowledge (Refer to Fig. 1.1).

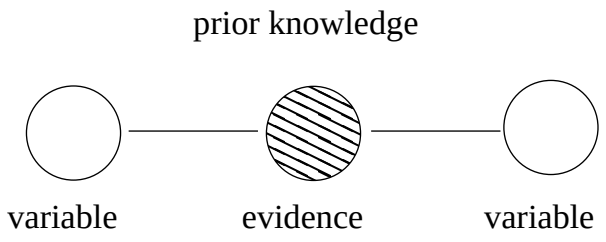

Figure 1.1: Word aligners by Brown et al. and Vogel et al. do not have much extensibility, while the graphical model implementation of the word aligner is extendible. This figure shows the situation where the prior knowledge is incorporated in a word aligner.

The second motivation comes from the linguistic level. Since SMT is designed on the *pair assumption*, it is quite natural that if something exists which would break the pair assumption it will effect the overall performance. Table 1.1 shows four such examples which break the pair assumption. In the first and the second examples, the English side includes some additional words,

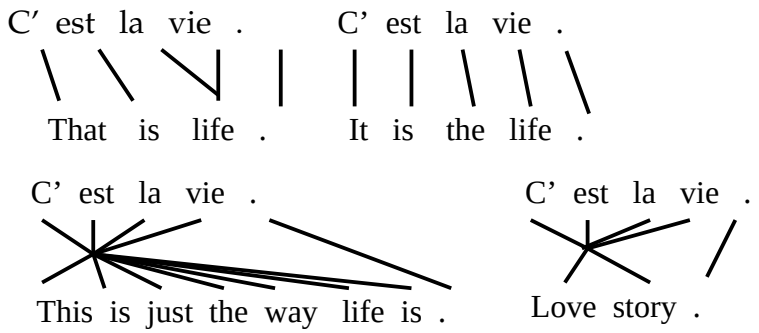

Figure 1.2: Examples of many-to-many mapping objects. The lower row include two many-to-many mapping objects, e.g. 4-to-7 and 4-to-2 mapping objects.

‘particular’ and ‘actually’. These are called *translational noise*. If we focus on Japanese or Chinese, the situation is much worse. Indeed, we come across varieties of such translational noise very easily. The third example shows an example of where different syntactic structures are used. In general, we call some objects which are inherently difficult to map literally as defined in the word alignment level *many-to-many mapping objects*. The fourth example shows the necessity to consider semantics.

Among these, one of our targets is many-to-many mapping objects. Figure 1.2 depicts the situation where many-to-many mapping objects are yielded quite naturally in the context of human translation: human translators can translate using whatever text they consider conveys the meaning of the source text. Among them, consider the sentence pairs (*‘c’ est la vie’*, ‘that is life’) and (*‘c’ est la vie’*, ‘love story’). In these cases, even native speakers will not be able to align the source and target words. By the architecture of IBM Model-based word alignment, it is known that such objects have a potential danger of not being extracted in the process, although many such words still have the possibility to be aligned correctly by phrase extraction heuristics in the translation model (Och and Ney, 2003).

Conversely, we prepare a toy situation where many-to-many mapping objects exist in the parallel corpus in Figure 1.3. For example, we can consider ‘to my regret’, ‘i am sorry that’, ‘it is a pity that’, and ‘sorry’ as many-to-many mapping objects (or paraphrases). Similarly in this context,

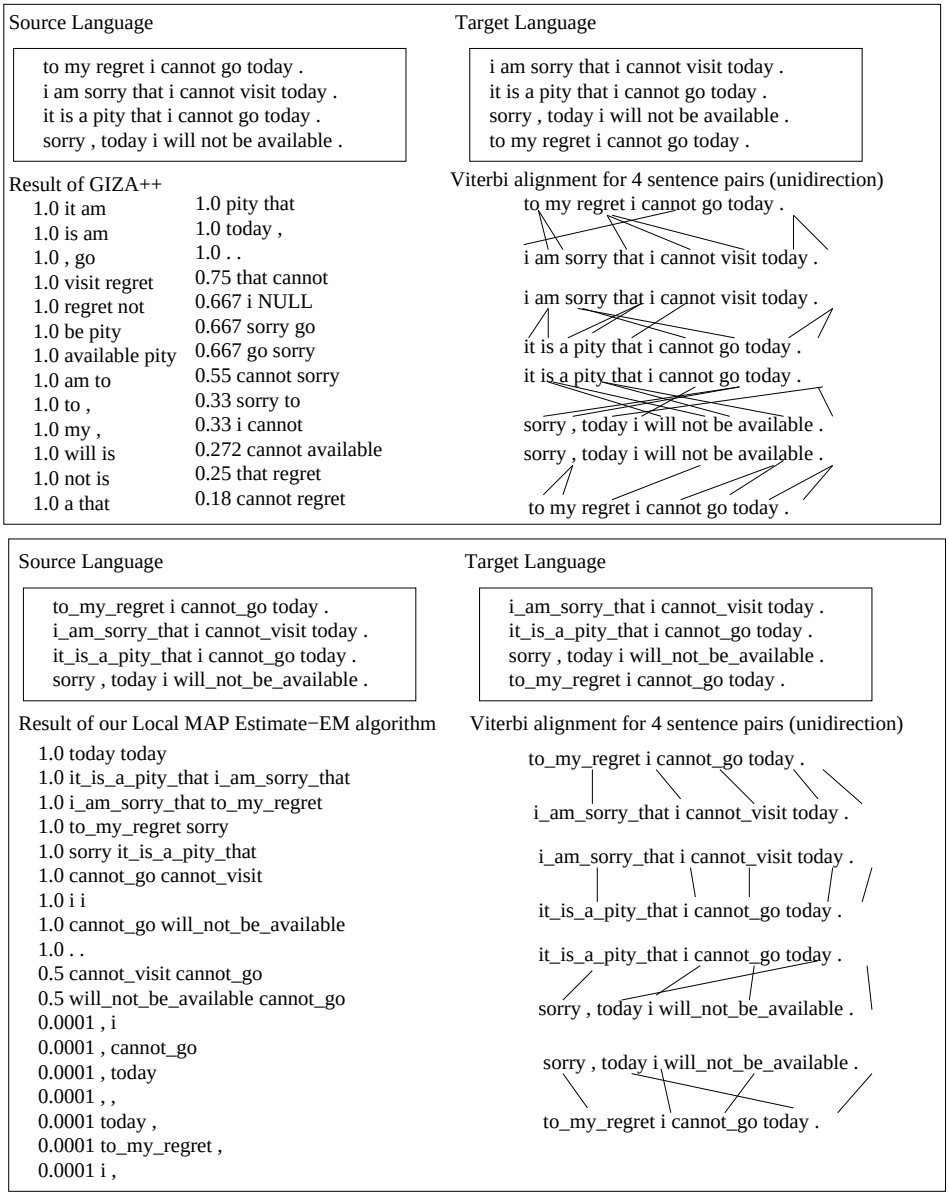

Figure 1.3: An example alignment of paraphrases (In the example both source and target language is English for readability.) in which the training corpus consists of four sentence pairs. (Upper figure): Results show that only the matching between the colon is correct (See the second row in the rightmost column). Note that the matching between “is” and “am” is close (See the fourth row in the leftmost column). (Lower figure): An example alignment of paraphrases by our Local MAP Estimate-EM algorithm. As a prior knowledge, we incorporate the anchor words shown in Table A.1.

sentence ID	target	source	position in tgt	position in src
1	to_my_regret	i_am_sorry_that	1	1
1	i	i	2	2
1	cannot_go	cannot_visit	3	3
1	today	today	4	4
1	.	.	5	5
2	i_am_sorry_that	it_is_a_pity_that	1	1
2	i	i	2	2
2	cannot_visit	cannot_go	3	3
2	today	today	4	4
2	.	.	5	5
3	it_is_a_pity_that	sorry	1	1
3	i	i	2	4
3	cannot_go	will_not_be_available	3	5
3	today	today	4	3
3	.	.	5	6
4	sorry	to_my_regret	1	1
4	i	i	4	2
4	will_not_be_available	cannot_go	5	3
4	today	today	3	4
4	.	.	6	5

Table 1.2: Example of biterm correspondence which is given to the Local MAP Estimate-EM aligner.

'cannot go', 'cannot visit', and 'will not be available' are other many-to-many mapping objects (or paraphrases). If human beings were to align these, one way of doing so would be as shown on the righthand side of the lower figure. However, as is shown in the upper figure, the result of GIZA++ includes a number of wrong alignment links.[3] In fact, the lower figure does not show the result of human analysis, but of our MAP-based aligner (Refer to Section 3.5.3, 3.5.4, etc; We gave the prior knowledge about alignment links as in Table 1.2). This example shows that our MAP-based word aligner overcomes this to derive a solution in this case.[4] Firstly, it is noted that even if a parallel

[3]It is noted that a traditional word aligner often assumes that a fairly big parallel corpus is given. In this sense, it might be a mistake from the beginning to consider to use a traditional word aligner in this case. However, we try to let this corpus fairly simple to be aligned: the length of four sentences are quite similar (9, 10, 9, and 8) and among 35 words they include 5 times of 'i', 4 times of 'today' and '.', 3 times of 'cannot', and twice of 'go'. This will make both the model complexity and the data complexity are small. Nevertheless, GIZA++ fails in aligning this.

[4]We noticed that there are two commas after 'sorry' in this parallel corpus. These commas cause the probability

corpus includes many-to-many mapping objects, there are many cases where GIZA++ aligns them correctly mainly due to the following process of symmetrization or the phrase extraction heuristics. Secondly, a traditional word aligner assumes that a fairly big parallel corpus is given. In practice, we may be faced with a tiny corpus as in this case. In this sense, our goal is (1) to achieve the performance even if a corpus is contaminated with bad many-to-many mapping objects, and (2) to provide a method which has scalability from tiny data to big data.

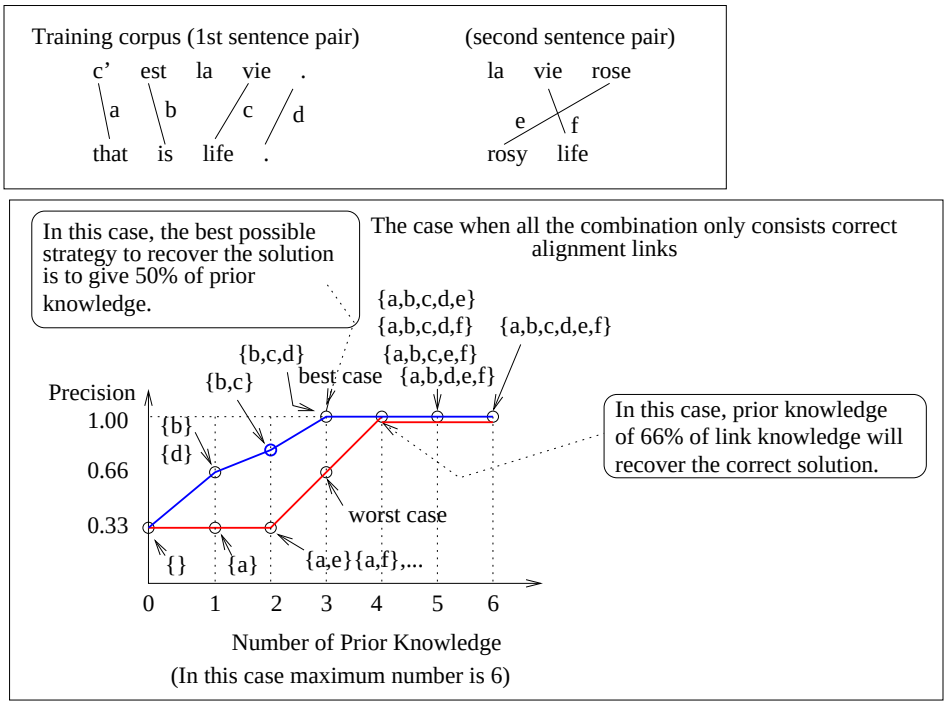

Figure 1.4: We show how much information about alignment links was required to recover the precision which is shown in the y-axis. If we gave more than four correct alignment links, the MAP-based aligner was able to obtain the correct alignment. If we gave three correct alignment links, the solution was correct in the case of $\{b, c, d\}$. However, for other cases such as $\{a, e, f\}$, the precision was 0.66. The point at 0 in the x-axis indicates the performance of a traditional word aligner where no prior knowledge was provided. The precision was 0.33.

Once such a MAP-based aligner is built, our interest is to show how to use this word aligner. Firstly, all the information about alignment links is provided to the MAP-based aligner in this case as is shown in Table 1.2. Since the aim of traditional word aligners is to obtain information

0.001. If we happen to remove these commas, we obtained the result as in Figure A.1 in Appendix.

about alignment links, the story seems somewhat upside down at first sight. However, this is our correct story. Our aim is to supply information about alignment links by other methods than word alignment. Our first example is bilingual noun phrase correspondences (Sections 3.9.4 and 3.10.3). If we extract bilingual noun phrases, we will know the alignment links between them. Similarly, we could use other linguistic knowledge to extract such bilingual correspondences (Details are explained in Section 3.9).

Secondly, however, it is, of course, not possible in practice to provide all the information about alignment links. (If this were possible, we would not need word alignment.) However, the good news is that if we know around 50-60% of such alignment links, the MAP-based aligner can be expected to obtain the alignment links successfully. Table 1.4 shows a schematic figure for a toy example. If we can give the information about three links $\{b, c, d\}$ among six links $\{a, b, \cdots, f\}$, the precision reaches 1.0 for this particular situation.

1.1 The Structure of the Thesis

This thesis is organized in the following way:

Chapter 2 gives a brief introduction of SMT and graphical models.

Chapter 3 presents our word aligner. In the sections on algorithmic design (3.4–3.6), Sections 3.4 and 3.5 discuss how to learn from a parallel corpus, while Section 3.6 discusses inference. These three sections give the foundation of our MAP-based word aligner. The first section presents the model without Markov dependencies and the second section explains the HMM Model. We use such MAP-based word aligners as a tool to investigate the aspect of noise throughout the word alignment chapter. Section 3.7 Data Design examines methods to investigate the aspect of data manipulation. Section 3.8 Linguistic Domain Knowledge about Word Alignment Links explores the relations among variables (This is often called linguistic domain knowledge). Section 3.9 Experiments provides our results.

Chapter 4 gives a (hierarchical) Pitman-Yor process-based language modelling and translation modelling.

Chapter 5 concludes together with some avenues for further research.

1.2 Contributions of this Thesis

The summary of the contributions of this thesis is as follows.

1. The proposal of local MAP estimate-EM and MAP assignment-EM algorithms (Sections 3.4.3 and 3.4.4).

2. Sentence-level outlier detection algorithm / word-level noise sensitive MAP-based word aligner (Sections 3.7.2 and 3.7.3).

3. Application of (hierarchical) Pitman-Yor process to language model and translation model in the context of Machine Translation (Sections 4.1.2, 4.1.3 and 4.2.1).

1.3 Notation

We use the following notation throughout this thesis. For the description of a Pitman-Yor process,

e	an English word
f	a foreign word
$\breve{e}$	an English sentence
$\bar{e}$	an English phrase
$\lvert\breve{e}_i\rvert$	the length of sentence $\breve{e}_i$
$\breve{e}_{\bar{i}}$	a reference translation of foreign sentence $\breve{f}_i$
$P(e\lvert f)$	a lexical translation probability (or T-table) for word e over word f
$P(\bar{e}\lvert\bar{f})$	a translation model for phrase $\bar{e}$ over $\bar{f}$
$P_{LM}(e)$	a language model for e
$P_{LM}(\bar{e})$	a language model whose segmentation follows $\bar{e}$
a	an alignment function

we use the following notation:

$c(n)$	count of events n
$PY(d, \theta, G)$	Pitman-Yor process with discount parameter d, strength parameter θ and base distribution G
u	context
$G_\emptyset \sim PY(d_0, \theta_0, G_0)$	a distribution $G_\emptyset$ has the underlying distribution $PY(d_0, \theta_0, G_0)$

1.4 Related Publications

This thesis is based on the following eleven publications.

- Tsuyoshi Okita. Data cleaning for word alignment. *In Proceedings of Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2009) Student Research Workshop*, pages 72–80, 2009.

- Yanjun Ma, Tsuyoshi Okita, Ozlem Cetinoglu, Jinhua Du, and Andy Way. Low-resource Machine Translation using MaTrEx: the DCU Machine Translation system for IWSLT 2009. *In Proceedings of the International Workshop on Spoken Language Translation (IWSLT 2009)*, pages 29–36, 2009.

- Tsuyoshi Okita. Preprocessing method for word alignment. *CLUKI colloquim, DCU, Dublin*, 2009.

- Tsuyoshi Okita, Alfredo Maldonado Guerra, Yvette Graham, and Andy Way. Multi-Word Expression-sensitive word alignment. *In Proceedings of the Fourth International Workshop On Cross Lingual Information Access (CLIA2010, collocated with COLING2010), Beijing, China.*, 2010.

- Tsuyoshi Okita, Jie Jiang, Rejwanul Haque, Hala Al-Maghout, Jinhua Du, Sudip Kumar Naskar, and Andy Way. MaTrEx: the DCU MT System for NTCIR-8. *In Proceedings of the*

NII Test Collection for IR Systems-8 Meeting (NTCIR-8), Tokyo., pages 377–383, 2010.

- Tsuyoshi Okita, Yvette Graham and Andy Way. Gap between theory and practice: Noise sensitive word alignment in machine translation. *In Proceedings of the Workshop on Applications of Pattern Analysis (WAPA2010). Cumberland Lodge, England.*, 2010.

- Tsuyoshi Okita and Andy Way. Hierarchical Pitman-Yor Language Model in Machine Translation. *In Proceedings of the International Conference on Asian Language Processing (IALP 2010), Harbin, China, December*, 2010. pp.245-248.

- Tsuyoshi Okita and Andy Way. Statistical Machine Translation with Terminology. *The First Symposium on Patent Information Processing (SPIP), Tokyo, Japan*, Dec, 2010. pp. 41-47.

- Tsuyoshi Okita, Alexandru Ceausu, and Andy Way. Statistical Machine Translation with Factored Translation Model: MWEs, Separation of Affixes, and Others. Poster presentation. *In Proceeding of the 24th Florida Artificial Intelligence Research Society Conference (FLAIRS-24). Palm Beach, Florida*. pp. 353-354.

- Tsuyoshi Okita and Andy Way. Given Bilingual Terminology in Statistical Machine Translation: MWE-sensitve Word Alignment and Hierarchical Pitman-Yor Process-based Translation Model Smoothing. *In Proceedings of the 24th International Florida Artificial Intelligence Research Society Conference (FLAIRS-24). Palm Beach, Florida*. pp.269-274.

- Tsuyoshi Okita and Andy Way. Pitman-Yor Process-based Language Model. *International Journal of Asian Language Processing (IJALP), 21(2)*, pp.57-70, 2011.

Chapter 2

Background Knowledge

2.1 Parameter Estimation in Statistical Machine Translation

A classical formulation of PB-SMT in an end-to-end setting, i.e. Bayesian noisy channel model, can be written as in Definition 1.

Definition 1 (Bayesian Noisy Channel)**.** *We assume that sentence pairs* $(\breve{e}, \breve{f})$ *are drawn i.i.d. (independent and identically distributed) according to the fixed (but unknown) underlying distributions* $p(\bar{f}|\bar{e})\cdot p(e)$*. Then, for a given test sentence* $\breve{f}$*, our task is to obtain a sentence* $\underline{\breve{e}}$ *which maximizes the following problem (2.1):*

$$\begin{cases} \underline{\breve{e}} = & \arg\max_e p(\bar{f}|\bar{e}) \cdot p_{LM}(\bar{e}) & \textit{(decoding task)} \\ & \textit{such that} \begin{cases} |\hat{p}(\bar{f}|\bar{e}) - p(\bar{f}|\bar{e})| \leq \delta_1 & \textit{(translation modeling task)} \\ |\hat{p}(\bar{e}) - p(\bar{e})| \leq \delta_2 & \textit{(language modeling task)} \end{cases} \end{cases} \tag{2.1}$$

where $p(\bar{f}|\bar{e})$ *denotes the target probability of the phrase alignment task,* $p(e)$ *denotes the target probability of the language modeling task (up to Markov order* n*; typical* n *is around 5),* $\hat{p}(\bar{f}|\bar{e})$ *and* $\hat{p}(\bar{e})$ *denote the true probability, and* $||$ *denotes some distance measure between two probability densities where* δ_1 *and* δ_2 *are some small quantities near zero.*

In order to achieve better performance, it becomes quite common to accommodate multiple translation models (Dyer et al., 2008) and language models. Note that since the storage performance in SMT affects the performance in training and decoding, it is common to use various compact formats to represent data which achieve small heap footprint, such as a lattice, a hypergraph, a confusion network, and a forest.

2.1.1 Translation Model

Refer to our notation defined in Section 1.2 in reading Definition 2 below.

Definition 2 (Word Alignment Task)**.** *Let $\breve{e}_i$ be the i-th sentence in the target language, $e_{i,j}$ be the j-th word in the i-th sentence, and e_i be the i-th word in the parallel corpus (Similarly for $\breve{f}_i$, $f_{i,j}$, and f_i). We are given a pair of sentence-aligned bilingual texts $S = \{(\breve{f}_1, \breve{e}_1), \ldots, (\breve{f}_n, \breve{e}_n)\}$, where each sentence can be composed of segments of phrases, that is $\breve{f}_i = (\bar{f}_{i,1}, \ldots, \bar{f}_{i,|f_i|})$ and $\breve{e}_i = (\bar{e}_{i,1}, \ldots, \bar{e}_{i,|e_i|})$. For a given word pair (e, f), the task of word alignment is to find a lexical translation probability $p_{f_i} : e_i \rightarrow p_{f_j}(e_i)$ such that $\Sigma p_{f_j}(e_i) = 1$ and $\forall e_i : 0 \leq p_{f_j}(e_i) \leq 1$ (It is noted that some models such as IBM Models 3 and 4 have deficiency problems[1]). It is noted that there may be several words in the source language and the target language which do not map to any words; these are called unaligned (or null aligned) words. Triples $(\bar{f}_i, \bar{e}_i, p_{\bar{f}_i}(\bar{e}_1)))$ are called T-tables.*

In practice, the IBM Models introduce an alignment (relative / absolute distortion) function [2] as a latent variable to solve this problem as a missing value problem. Note that the subsequent phrase extraction process assumes that a word alignment process yields word-aligned sentence pairs via Viterbi decoding, as in Definition 3.

[1] Deficiency problems mean that the sum of probabilities is not 1.

[2] An alignment function and a distortion function essentially refer to a similar idea. An alignment function is defined in IBM Models 1 and 2, while a distortion function is defined in IBM Model 3, 4 and 5. The role of an alignment function is to map the position of the foreign word into the position of English word.

Definition 3 (Word Alignment Task — Viterbi Decoding)**.** *For a given* (e, f)*, the task of word alignment is to find the most likely word-aligned sentence pairs.*

Based on such word-aligned sentence pairs, the phrase extraction process, shown in Definition 4, is invoked.

Definition 4 (Phrase Extraction)**.** *The phrase extraction algorithm extracts all consistent phrase pairs from a word-aligned sentence pair (Och and Ney, 2003).*

The idea of phrase extraction is to loop over all possible English phrases and find the minimal foreign phrase that matches each of them in principle.[3] Matching is done by identifying all alignment points for the English phrase and finding the shortest foreign phrase that includes all the foreign counterparts for the English words.

2.1.2 Language Model

Let w_i denote a word, and W denotes a sequence of words $w_1, w_2, \ldots, w_n$. A language model aims at modelling $p(W)$ (= $p(w_1, \ldots, w_m)$) such that $p(W)$ predicts the probability of picking up a sequence of words W. In an n-gram language model, the probability $p(w_1, \ldots, w_m)$ of observing the sentence $w_1, \ldots, w_m$ is approximated as in (2.2):

$$\begin{aligned} p(W) &= p(w_1, \ldots, w_m) \\ &= \prod_{i=1}^{m} p(w_i | w_1, \ldots, w_{i-1}) \\ &= \prod_{i=1}^{m} p(w_i | w_{i-m}, \ldots, w_{i-1}) \end{aligned} \tag{2.2}$$

Note that $p(w_1, \ldots, w_m) = \prod_{i=1}^{m} p(w_i | w_1, \ldots, w_{i-1})$ holds by the chain rule to express the joint distribution for a sequence of observations, while $P(w_n | w_1, \ldots, w_{n-1}) = P(w_n | w_{n-m}, \ldots, w_{n-2}, w_{n-1})$ holds by the Markov assumption of the history up to m words.

[3] Consistency always requires that these foreign words do not align to other English words.

The measure to evaluate the performance of a language model is often done by perplexity defined as in (2.3):

$$2^{H(P_{LM})} \tag{2.3}$$

where the cross-entropy $H(P_{LM})$ is defined as in (2.4):

$$H(P_{LM}) = -\frac{1}{n}\sum_{i=1}^{n} \log P_{LM}(w_i|w_1, \ldots, w_{i-1}) \tag{2.4}$$

2.1.3 Overfitting

One characteristic of word alignment is shown on the right-hand side of Figure 2.1. The y-axis of both figures shows the class identity and the x-axis shows the number of words. The left-hand side of Figure 2.1 shows the case for POS tagging, while the right-hand side shows the case for word alignment. The maximum number on the x-axis on the left-hand side is around 50, while that on the right-hand side is around 1000. The discussion leads to the motivation in Chapter 3 of our smoothing method, specifically whether it may be useful to apply a power-law distribution-based smoothing method.

Modern Machine Learning algorithms, such as Support Vector Machines (Boser et al., 1992; Vapnik, 1998), seek to obtain a small generalisation error over unseen data. This mechanism is implemented by minimizing both the risk and the capacity of the function class. Suppose that we cannot automatically adjust the generalisation error which is often the case in Bayesian Machine Learning.[4] We start with some model complexity and we adjust this model complexity for some given data in order to obtain the best generalisation error over unseen data. In this context, if the initial model complexity is below the point which achieves the best generalisation error (or an equi-

[4]In Bayesian Machine Learning, An Information Criterion (or Akaike Information Criterion; AIC) (Akaike, 1974), Bayesian Information Criterion (BIC) (Schwarz, 1978), and Minimal Description Length (MDL) (Rissanen, 1978) are often used for measuring the model complexity.

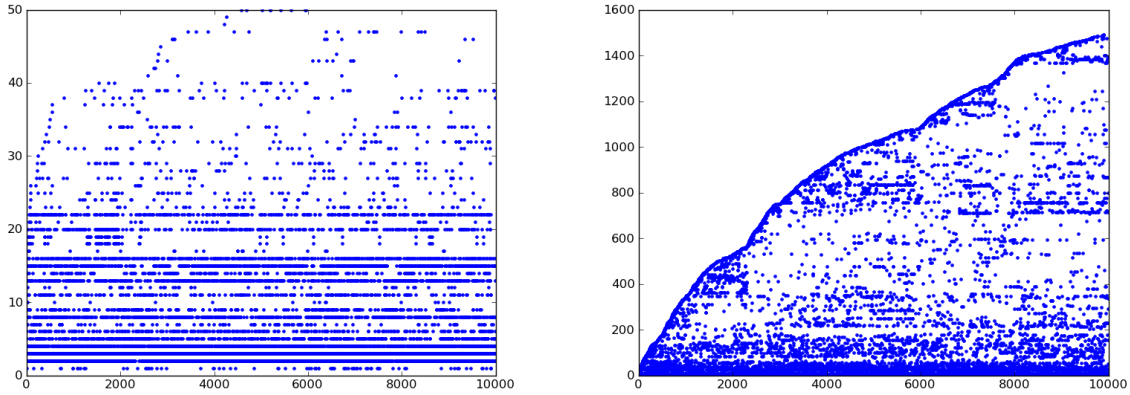

Figure 2.1: The left figure shows the POS tagging and the right figure shows the word alignment. In both figures, the class identity is shown in y-axis and the number of words is shown in x-axis.

librium point), this is called 'underfitting' (point A3 in Figure 2.2). If the initial model complexity is beyond the point which achieves the best generalisation error, this is called 'overfitting' (point A4 in Figure 2.2). A model selection technique aims at transfering A3 or A4 into an equilibrium state at A2. Analogous to this, we may apply the same idea to smoothing techniques where we aim at controlling the horizontal axis to transfer the state into an equilibrium state at A2.

2.2 Graphical Models

This section gives a brief overview of graphical models, especially two of the main algorithms on graphical models: sum-product and max-product algorithms. The description in this chapter follows (Bishop, 2006; Koller and Friedman, 2009).

2.2.1 Factor Graph

A factor graph is a generalization of Bayesian and Markov networks, as shown in (2.5):

$$p(\mathbf{x}) \quad = \quad \prod_{s} f_s(\mathbf{x}_s) \tag{2.5}$$

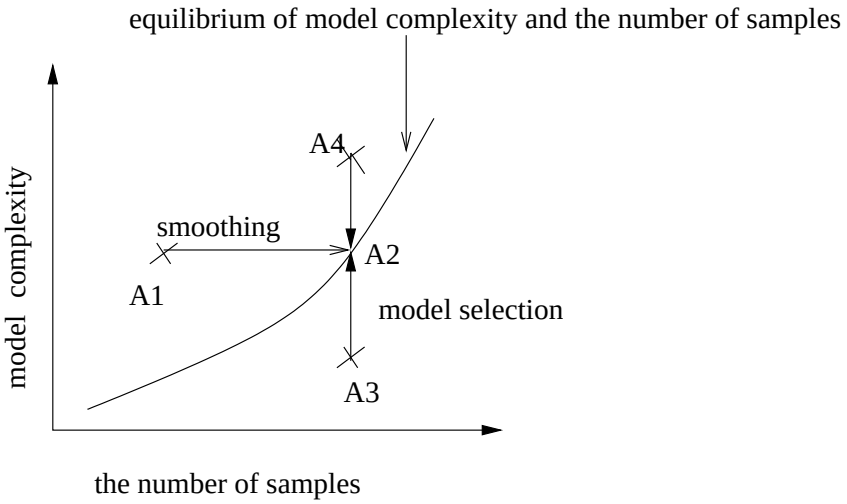

Figure 2.2: A figure explain our usage of the term 'underfitting'.

where $\mathbf{x}_s$ denotes a subset of variables and f_s is a factor which corresponds to a set of variables $\mathbf{x}_s$. Firstly, a factor graph is a generalization of a Bayesian network, which can be explained by the fact that the joint distribution of Bayesian network with K nodes can be written as in (2.6):

$$p(x_1, \ldots, x_K) = \prod_{k=1}^{K} p(x_k|\mathbf{pa}_k) \tag{2.6}$$

where $\mathbf{pa}_k$ denotes the set of parents of x_k. The fact that the joint distribution defined by a graph is given by the product of a conditional distribution for each node conditioned on the variables corresponding to the parents of that node in the graph, is called the factorization property for Bayesian network. Secondly, a factor graph is a generalization of a Markov network, which can be explained by the fact that the joint distribution of a Markov network over the maximal cliques of the graph can be written as a product of potential functions $\phi_C(x_C)$ as in (2.7):

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C} \phi_C(\mathbf{x}_C) \tag{2.7}$$

where Z is a normalization constant shown as in (2.8):

$$Z = \sum_{\mathbf{x}} \prod_{C} \phi_C(\mathbf{x}_C). \tag{2.8}$$

Now, we describe algorithms only for a factor graph, which as we have just shown can be used both for Bayesian and Markov networks. We describe two algorithms which are representative algorithms in graphical models for a tree-structured topology, which can be easily modified even when the topology of the graphical model changes. The sum-product algorithm aims at marginalization by performing sums, while the max-product algorithm aims at finding the values that maximize the marginals. For learning HMMs the sum-product algorithm is employed, while for decoding the max-product algorithm is employed.

2.2.2 Sum-Product algorithm

On the one hand, the marginal can be obtained by summing the joint distribution over all variables except x as in (2.9):

$$p(\mathbf{x}) = \sum_{\mathbf{x}\backslash x} p(x) \tag{2.9}$$

where $\mathbf{x}\backslash x$ denotes the set of variables in $\mathbf{x}$ except x. On the other hand, if the factor graph is in a tree structure, the joint distribution can be written as a product of the factor. Let f_s denote the factor node, X_s denote the set of all variables in the subtree connected to the variable node x through the factor node f_s, $F_s(x, X_s)$ denote the product of all the factors in the group associated with factor f_s, and $ne(x)$ denote the set of factor nodes that are neighbours of x. Then, the joint distribution can be written as a product of all the factors except x as in (2.10):

$$p(\mathbf{x}) = \prod_{s \in ne(x)} F_s(x, X_s) \tag{2.10}$$

From (2.9) and (2.10), the interchanging of sums and products yields (2.11):

$$\begin{aligned} p(\mathbf{x}) &= \prod_{s\in ne(x)} \left[\sum_{X_s} F_s(x, X_s)\right] \\ &= \prod_{s\in ne(x)} \mu_{f_s\to x}(x) \end{aligned} \tag{2.11}$$

Now we define two kinds of messages: from the factor nodes f_s to the variable node x, and from the variable node x_m to the factor nodes f_s as in (2.12):

$$\begin{cases} \mu_{f_s\to x}(x) &= \sum_{X_s} F_s(x, X_s) \\ \mu_{x_m\to f_s}(x_m) &= \sum_{X_{sm}} G_m(x_m, X_{sm}) \end{cases} \tag{2.12}$$

Then, we consider to compute the marginal $p(x)$ by the product of all the incoming messages arriving at node x as in (2.13) for F_s, and (2.14) for G_m:

$$F_s(x, X_s) = f_s(x, x_1, \ldots, x_M)G_1(x_1, X_{s1}), \ldots, G_M(x_M, X_{sM}) \tag{2.13}$$

$$G_m(x_m, X_{sm}) = \prod_{l\in ne(x_m)\setminus f_s} F_l(x_m, X_{ml}) \tag{2.14}$$

Substitute (2.13) into (2.12) yields (2.15) for F_s:

$$\begin{aligned} \mu_{f_s\to x}(x) &= \sum_{x_1}\cdots\sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m\in ne(f_s)\setminus x} \left[\sum_{X_{xm}} G_m(x_m, X_{sm})\right] \\ &= \sum_{x_1}\cdots\sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m\in ne(f_s)\setminus x} \mu_{x_m\to f_s}(x_m) \end{aligned} \tag{2.15}$$

a^1	a^1	a^1	a^1	a^2	a^2	a^2	a^2	a^3	a^3	a^3	a^3
b^1	b^1	b^2	b^2	b^1	b^1	b^2	b^2	b^1	b^1	b^2	b^2
c^1	c^2	c^1	c^2	c^1	c^2	c^1	c^2	c^1	c^2	c^1	c^2
0.25	0.35	0.08	0.16	0.05	0.07	0.0	0.0	0.15	0.21	0.09	0.18

$\Downarrow$

a^1	a^1	a^2	a^2	a^3	a^3
c^1	c^2	c^1	c^2	c^1	c^2
0.33	0.51	0.05	0.07	0.24	0.39

Table 2.1: Example of factor marginalization. Factor b is marginalized out by sum-product algorithm

Substitute (2.14) into (2.12) yields (2.16) for G_m:

$$\begin{aligned} \mu_{x_m \to f_s}(x_m) &= \prod_{l \in ne(x_m) \setminus f_s} \left[\sum_{X_{ml}} F_l(x_m, X_{ml}) \right] \\ &= \prod_{l \in ne(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m) \end{aligned} \tag{2.16}$$

Note that we use the initialized messages at the root of the tree and at the leaf nodes as in (2.17):

$$\begin{cases} \mu_{x \to f}(x) &= 1 \\ \mu_{f \to x}(x) &= f(x) \end{cases} \tag{2.17}$$

An easy example is shown in Table 2.1. For example, $P(a^1, c^1) = \sum [P(a^1, b^1, c^1), P(a^1, b^2, c^1)] = \sum [0.25, 0.08] = 0.33$.

2.2.3 Max-Product (Max-Sum) Algorithm

The aim of max-sum algorithm is to find the set of values of $\mathbf{x} = x_1^*, \ldots, x_M^*$ that *jointly* (not individually) maximizes the joint distribution $p(\mathbf{x})$ as in (2.18):

$$\max_{\mathbf{x}} p(\mathbf{x}) \quad = \quad \max_{x_1} \ldots \max_{x_M} p(\mathbf{x}) \tag{2.18}$$

The difference between *individually* and *jointly* will result in a different conclusion. Take an easy example. Consider the joint distribution $p(x_1, x_2)$: $p(0,0) = 0.3$, $p(0,1) = 0.3$, $p(1,0) = 0.4$, $p(1,1) = 0.0$. The result of joint maximization is 0.4 (at $p(1,0)$), while the indivual maximization is 0.3 (at $p(0,0)$) ($x_1 = 0$ for maximization with regard to x_1 and $x_2 = 0$ with regard to x_2).

Let M be the total number of variables. Suppose that a graph is in a chain form. In this case, the maximization of $p(\mathbf{x})$ can be written as in (2.19):

$$\begin{aligned} \max_{\mathbf{x}} p(\mathbf{x}) &= \frac{1}{Z} \max_{x_1} \ldots \max_{x_N} \left[\phi_{1,2}(x_1, x_2) \ldots \phi_{N-1,N}(x_{N-1}, N)\right] \\ &= \frac{1}{Z} \max_{x_1} \left[\phi_{1,2}(x_1, x_2) \left[\ldots \max_{x_N} \phi_{N-1,N}(x_{N-1}, N)\right]\right] \end{aligned} \tag{2.19}$$

We consider the messages sent from the leaves to the root. Similarly with the results of the sum-product algorithm of (2.15) and (2.16), we replace $\sum$ with $\max$ as in (2.20):

$$\begin{cases} \mu_{f \to x}(x) &= \max_{x_1, \ldots, x_M} \left[\ln f(x, x_1, \ldots, x_M) + \sum_{m \in ne(f_s) \setminus x} \mu_{x_m \to f}(x_m)\right] \\ \mu_{x \to f}(x) &= \sum_{l \in ne(x) \setminus f} \mu_{f \to x}(x) \end{cases} \tag{2.20}$$

Note that we use the logarithm in order to deal with the numerical underflow. (For this reason, the max-product algorithm is also called the max-sum algorithm. These two are essentially the same

except for the representation using the logarithm.) Initial messages are similarly given as in (2.21):

$$\begin{cases} \mu_{f\to x}(x) &= 0 \\ \mu_{x\to f}(x) &= \ln f(x) \end{cases} \tag{2.21}$$

Then, at the root node the maximum probability can be obtained as in (2.22):

$$p^{max} = \max_x \left[\sum_{s\in ne(x)} \mu_{f_s\to x}(x) \right] \tag{2.22}$$

Now, we consider the problem of finding the configuration of the variables, cf. Viterbi algorithm. Firstly, we consider the following problem to determine the most probable configuration as in (2.23):

$$x^{max} = \arg\max_x \left[\mu_{f_s\to x}(x)\right] \tag{2.23}$$

For this purpose, we send messages from the root to leaves applying (2.20) with (2.23):

$$\begin{cases} \mu_{x_n\to f_{n,n+1}}(x_n) &= \mu_{f_{n-1,n}\to x_n}(x_n) \\ \mu_{f_{n-1,n}\to f_x}(x_n) &= \max_{x_{n-1}} \left[\ln f_{n-1,n}(x_{n-1}, x_n) + \mu_{x_{n-1}\to f_{n-1,n}(x_n)}\right] \end{cases} \tag{2.24}$$

An initial message is shown in (2.25):

$$\mu_{x_1\to f_{1,2}}(x_1) = 0 \tag{2.25}$$

Then, the resulting x_N which is the most probable value can be obtained by (2.26):

$$x_N^{max} = \arg\max_{x_N} \left[\mu_{f_{N-1,N}\to x_N}(x_N)\right] \tag{2.26}$$

a^1 b^1 c^1	a^1 b^1 c^2	a^1 b^2 c^1	a^1 b^2 c^2	a^2 b^1 c^1	a^2 b^1 c^2	a^2 b^2 c^1	a^2 b^2 c^2	a^3 b^1 c^1	a^3 b^1 c^2	a^3 b^2 c^1	a^3 b^2 c^2
0.25	0.35	0.08	0.16	0.05	0.07	0.0	0.0	0.15	0.21	0.09	0.18

$\Downarrow$

a^1 c^1	a^1 c^2	a^2 c^1	a^2 c^2	a^3 c^1	a^3 c^2
0.25	0.35	0.05	0.07	0.15	0.21

Table 2.2: Example of factor marginalization. Factor b is marginalized out by max-sum algorithm

Secondly, we determine the state sequence corresponding to the most probable configuration. This problem can be solved by keeping track of which values of the variables yields the maximum state of each variable which is shown in (2.27):

$$\phi(x_n) \quad = \quad \arg\max_{x_{n-1}} \left[\ln f_{n-1,n}(x_{n-1}, x_n) + \mu_{x_{n-1}\to f_{n-1,n}}(x_n)\right] \tag{2.27}$$

Once the most probable value of the final node x_N is obtained, we follow the link back to find the most probable state of node x_{N-1}, which is repeated until the initial node x_1. This is called back-tracking, which is shown as in (2.28):

$$x_{n-1}^{max} \quad = \quad \phi(x_n^{max}) \tag{2.28}$$

An easy example using the same data as in Table 2.1 is shown in Table 2.2. For example, $P(a^1, c^1) = \max\left[P(a^1, b^1, c^1), P(a^1, b^2, c^1)\right] = \max[0.25, 0.08] = 0.25$.

2.2.4 Typical Inferences

This section describes three most common query types following (Koller and Friedman, 2009; Murphy, 2007). Let $\mathbf{X} = \{X_1, \ldots, X_n\}$ be a set of random variables, and $P(X_1, \ldots, X_n)$ be the joint distribution over a set $\mathbf{X}$. Let us partition the variables $\mathbf{X}$ into $\mathbf{E}$ (evidence), $\mathbf{Q}$ (query), and

H (hidden / nuisance).

Let a subset E of random variables in the model denote the evidence, e denote its instantiation to these variables, a subset of Y of random network denote the query variables, and W denotes a (non-evidence) variables, that is $\mathbf{X} - E$.

- Conditional probability query (posterior):

$$P(X_{\mathbf{Q}}|x_{\mathbf{E}}) \quad \propto \quad \sum_{X_{\mathbf{H}}} p(X_{\mathbf{Q}}, x_{\mathbf{E}}, x_{\mathbf{H}}) \tag{2.29}$$

- MAP estimate when $\mathbf{H} = \emptyset$ (posterior mode):

$$x^*_{\mathbf{Q}} \quad = \quad \arg\max_{X_{\mathbf{Q}}} p(x_{\mathbf{Q}}|x_{\mathbf{E}}) = \arg\max_{X_{\mathbf{Q}}} p(x_{\mathbf{Q}}, x_{\mathbf{E}}) \tag{2.30}$$

- Marginal MAP estimate (mode of marginal posterior):

$$x^*_{\mathbf{Q}} \quad = \quad \arg\max_{X_{\mathbf{Q}}} p(x_{\mathbf{Q}}|x_{\mathbf{E}}) = \arg\max_{X_{\mathbf{Q}}} \sum_{X_{\mathbf{H}}} p(x_{\mathbf{Q}}, x_{\mathbf{E}}, x_{\mathbf{H}}) \tag{2.31}$$

Chapter 3

Word Alignment: Noise

This chapter aims at identifying *noise* in the level of word alignment in Statistical Machine Translation (SMT), as well as providing a prototype for handling such noise in word alignment. The aim of such identification is to improve the overall performance of the word aligner. We handle two kinds of noise: word-level noise and sentence-level noise. We present quite different methods for these two.

For word-level noise, we first introduce a novel MAP-based word aligner which has the capability of incorporating prior knowledge about alignment links as described in Chapter 1, and then through this interface we try to incorporate them. In this sense, one of our contributions is mainly the analysis, design, and development of this MAP-based word aligner. The main function of this MAP-based word aligner is, however, to incorporate knowledge not about *noise* itself, but about alignment links. Hence, the usage of this word aligner is to first detect various alignment links which may act as *noise* to the word aligner, and then to supply such information to a word aligner. In other words, many-to-many mapping objects may cause problems in word alignment. Instead, our idea is to extract a set of possible correct alignment links between many-to-many mapping objects (which is considered *noise*) before we run word alignment, and then to supply such information to the MAP-based word aligner. In this sense, before we run the MAP-based word aligner,

we need to identify which objects will potentially prove problematic in the word alignment process. We will present the up-to-date results including the lists of such possible candidates which become *noise* objects although this process to select candidates is not theoretical, but unfortunately empirical.

For sentence-level noise, we introduce a method which is similar to the outlier detection method in computer vision or in Machine Learning. If we trace the sentence-based training set errors, a word aligner will not work for the specific sentence pair. Coversely, if we assess the sentence-based training set errors, it would be possible to estimate the performance of the word aligner for the particular sentence pair. In this case, sentences that are trained badly by a word aligner are labeled with *noise*. In this case, we will not be able to classify *noise* objects since all the sentence pairs can equally be a candidate of *noise*.

Now, we mention briefly which section explains what topics. In Section 3.1, we review what is *noise* in audio, vision and word alignment. The term *noise* is often used in the context of audio and vision but has rarely been used in SMT. On the one hand, the crucial difference from audio and vision is that *noisy* objects may also work as a *signal* in SMT since all the fragments in a sentence usually have some meaning (Refer to the definition of *noise* in Section 3.1.3). On the other hand, the similarity is that it is often very difficult to detect such *noise* until we detect the *signal*, as mentioned in the above two algorithms. As is already mentioned, we cannot list a candidate of *noise* at the sentence-level; we only describe a list of potential candidates for *noise* at the word-level. This list starts with many-to-many mapping objects (*noise* as well as *signal*), translational noise (*noise* that is not a *signal*), and so forth.

Section 3.2 explains a hand-annotated corpus between JP-EN which we build. Then, Section 3.3 explains the method of evaluation. We use Alignment Error Rate (AER) (Och and Ney, 2003). For the sake of SMT, our aim is to achieve an overall better performance in terms of BLEU (Papineni et al., 2002). However, in terms of the quality of word alignment, it is not always clear if we only look at BLEU. Unfortunately, these two measures are not well correlated with each

other. Even if we obtain high-quality results in terms of AER, it may not always lead to better overall results in terms of BLEU (Fraser and Marcu, 2007). Nevertheless, it is still worthwhile to investigate word alignment in terms of AER for several reasons. Firstly, it is widely recognized that the IBM Models work quite well if we measure the performance by BLEU. This is true for most European languages as well as for language pairs comprising Asian languages and European languages, including EN-JP and EN-ZN. However, except for EN-FR we may not even know the actual performance on AER. This is due to the scarce availability of hand-annotated word alignment corpora. Even between EN-FR, most of the cases are measured using a Hansard annotation corpus of just 448 sentences (Och and Ney, 2003). For EN-JP, we could not find any report nor any hand-annotated corpus. Accordingly, it took us some time to build such a hand-annotated corpus between EN-JP, and to measure the performance. The first observation is that the AER from EN to JP is always several points higher than the other direction. Unlike word alignment between European languages, in the case of JP, we often split the input into words and morphemes using a morphological analyzer. Hence, we align morphemes (JP) to words (EN). If the source includes morphemes (JP), there is a strong possibility that such morphemes connect with content words, which is the case from JP to EN. Conversely, if the source includes only words (EN), even if we have morphemes in the target side there are less possibilities that morphemes connect with content words. Note that if we have to align these manually, there will be a low inter-annotator agreement whether we allow the alignment links between morphemes and content words or not. Section 3.4 explains a Bayesian Machine Learning approach where multiple solutions may be obtained.

From Section 3.5 to Section 3.7, various algorithms proposed for word alignment are explained. In order to investigate such *noise*, we do not look at the word aligner as a black box, but we try to understand the architectural weakness of the dominant word aligner GIZA++ (Och and Ney, 2003). As is mentioned above, one of the important tools we need was the MAP-based word aligner which incorporates prior knowledge into the word alignment process. We describe how to specify the prior knowledge into the process.

Section 3.8 considers the basic algorithm for data manipulation. After explaining the heuristic sentence-cleaning algorithm introduced in Moses, we introduce sentence-based and word-based cleaning algorithms. Section 3.9 mentions the characteristics of language. This section intends to obtain the relation between variables.

Then, in Section 3.10, we gather all the techniques — algorithms and data manipulations — in order to compute the overall performance as a word aligner.

3.1 Noise

Since it is not common to use the term *noise* in the context of SMT, this section describes what we mean by *noise*. We use this term *noise* as an analogy of *noise* in audio or vision. We discuss, however, that the characteristics of *noise* in SMT are quite different in nature. However, it turned out that rather than the pure *noise*, we need to take care of *signals* since such *signals* work as *noise* at the same time. All the more, the input of the MAP-based word aligner is word alignment links (not *noise*). Eventually, at the end of this section, we will give not a categorization of *noise* but a list of candidates which may provide the source of word alignment links. We start with the audio noise and the visual noise. Then, we proceed to word alignment.

3.1.1 Noise in Audio

In audio, *noise* is often defined alongside *signals*, but sometimes it is not. For example in the context of radio transmission and audio recording via records or tapes, *signals* may be the conversation of persons, music, and plays which we aim to transmit or record, while *noise* is the sound which disturbs our ability to hear *signals* and is often irrelevant to the *signals* themselves. High levels of noise can block, distort, change or interfere with the signals. As another example, suppose that our aim is to hear the conversation of other people. The conversation of other people is not particularly called as the *signals*, while the loud sounds which disburb us from hearing the conversations of the other people are called *noise*. Note that *noise* is often distinguished from *distortion* although those

two are alike. The latter is an unwanted alteration of the signal waveform.

Various noise reduction techniques are developed in order to reduce the former kinds of *noise*. There are several kinds of noise whose causes are different (Wikipedia[1], 2011). Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical : thermal noise (electronic noise), shot noise, flicker noise, burst noise, and avalanche noise. Thermal noise is due to the random thermal motion of electrons inside an electric conductor. (Electronic noise is the random variations in current or voltage caused by the random movement of electrons on the circuit.) Shot noise arises when the finite number of energy-carrying particles becomes significant among very low-level signals. Flicker noise, which arises in semiconductor devices, is a signal with a frequency spectrum that falls off steadily into the higher frequences, which occurs in almost all electronic devices. Burst noise is a sudden step-like transition between two or more levels, as high as several hundred microvolts, at random and unpredictable times. Avalanche noise arises when a junction diode is operated at the onset of avalanche breakdown.

3.1.2 Noise in Vision

Noise in vision is comparable with that in audio. *Noise* in vision is often defined as the information which we do not want to extract even if we manage to detect it. However at the same time, it is often the case that we do not know how to measure nor to extract *noise* in an image.

There are several kinds of *noise* known (Wikipedia[2], 2011): amplifier noise, salt-and-pepper noise, shot noise, quantization noise, film grain, and anisotropic noise. Amplifier noise is an additive stationary Gaussian noise (or white noise) caused primarily by thermal noise. This noise makes up a major part of the noise by image sensors. Salt-and-pepper noise (or spike noise) has a fat-tail distribution caused by analog-to-digital converter errors or bit transmission errors. Shot noise is the dominant noise in the lighter parts of an image from an image sensor, which is due to the statistical quantum fluctuations. Quantization noise is caused by quantizing the pixels of a

[1]Wikipedia, the free encyclopedia. `http://en.wikipedia.org`.
[2]Wikipedia, the free encyclopedia. `http://en.wikipedia.org`.

sensed image to a number of discrete levels. Film grain is a signal-dependent noise due to the grain of photographic film with a similar statistical distribution as shot noise. Anisotropic noise appears with a significant orientation in images. For example, image sensors are often subject to row noise or column noise.

One observation in computer vision is that it is often the case that even if we know the cause of noise in advance, it does not often help us how to construct the method to extract them. This is because *noise* is often the object which we do not know how to measure and to extract; we often extract them by extracting *signals*, then we first realize that the remaininder is *noise*.

3.1.3 Noise in Word Alignment in Statistical Machine Translation

The definition of *noise* in this section is intended only for word-level noise. Firstly, in contrast to audio and vision where most of the *noise* is not part of the *signal*, it turns out that many-to-many alignment objects work as *noise* as well as valid training data (hence *signal*) (Okita et al., 2010a). Secondly, in a sentence, the unit of *noise* may easily be changed (For example, some *noise* is defined in a word, but other *noise* is defined in a phrase.) In sum, our definition becomes as follows:

1. Any *fragments* may have (1) both *signals* and *noise*, (2) only *signals*, (3) only *noise*, and (4) neither *signals* nor *noise*.

2. *Segments* can overlap with other *segments*.

It is noted that such overlapped approach in translation model is investigated in Kaariainen (2009). However, this definition means that there is no obvious *noise*, while each *noise* type has a name in audio and vision. At the same time, it means that we are not capable of finding *noise* via detecting *signals*. Hence, this definition does not give us help.

The categorization here is not the categorization of either *noise* or *signal*, but rather word alignment links. For this reason, some types of noise have a name of noise, but others do not.

Firstly, this is due to the overlapping definition of *noise* and *signal* as is mentioned above. There is not much obvious *noise*, but most of them are *signals*. Secondly, as is explained at the begining of this chapter, the input of the MAP-based word aligner is not *noise*, but word alignment links, so it is no use if we categorize *noise*. What we need is a categorization based on word alignment links. Thirdly, we aim at only the word-level noise for word alignment. Fourthly, not all the objects may become noise due to the phrase extraction (For example, even though the unidirectional word aligner did not detect many-to-many mapping objects, a phrase extraction process may detect many-to-many mapping objects.) Hence, each item in the categorization below leaves open the possibility that these objects may become *noise*.

1. Many-to-many mapping object pairs (noun phrases, MWEs and paraphrases).
2. Translational noise.
3. Non-literal translation pairs.
4. Less frequent word pairs.
5. Human errors and typos (For example, ‘hmman’ will not align with ‘human’. Once errors and typos are corrected, we can align them correctly).
6. Anaphora and pronouns.

It should also write the condition when *noise* yields easily:

1. In the case where there are many possible correspondences between the source side and the target side.
2. In the case where the same symbol in the semantic level are expressed differently in the surface level (e.g. “(”, “[”,“<”). One common way is to normalize the surface forms in order to align them correctly. However, this is not a conclusive way to handle this correctly.

3. In the case where the same objects in the semantic level are expressed differently in the syntactic level (e.g. conjugations). Since the aim of word alignment is to align the corresponding verbs or nouns correctly, syntactic difference will be an obstacle to this.

4. In the case where the lengths in the source side and in the target side are different (Note that there are mechanisms such as fertility and NULL insertion in IBM Model 3 and 4 to deal with such cases).

5. In the case where a verb phrase is converted into a noun phrase, or vice versa.

It is noted that some of these will be investigated in Section 3.9.

3.2 Hand-Annotated Corpus-based Evaluation

We evaluate our methods in this chapter based on AER which requires a hand-annotated corpus. We used three kinds of corpus: the Hansard corpus between EN-FR (Och and Ney, 2003) and two other corpora which are built by us for EN-JP.

name	language pair	person	size	evaluation
Hansard	EN-FR	Och & Ney	484	AER
IWSLT	EN-JP	Okita	100	AER
NTCIR	EN-JP	Okita	50	AER
NTCIR	EN-JP	Fujii	3200k	BLEU
NTCIR	EN-JP	Fujii	200k	BLEU
NTCIR	EN-JP	Fujii	50k	BLEU
IWSLT	EN-JP		40k	BLEU

Table 3.1: The first three lines show the hand annotated corpora which we used in the evaluation by AER, while the four last lines show the corpora which we used in this thesis for evaluation by BLEU.

We constructed two kinds of corpus between EN-JP. The first corpus is the IWSLT 2006 subcorpus consisting of 100 sentence pairs (cf. Table 3.2). The second corpus is the NTCIR subcorpus of 50 sentence pairs. We use the alignment process of Lambert et al. (Lambert et al., 2006).

The main difference is that the resulting variety of opinions regarding the correctness of the links is expected to be so wide in the case of the EN-JP corpus that we may need 5 or 6 persons in order to obtain the same effect as in the EN-FR corpus. We take the approach not to give the average of many persons, but rather to adopt one annotation which is consistent throughout the corpus.[3]

```
# Sentence pair (5) source length 9 target length 10
we want to have a table near the window .
NULL ({ 1 2 5 }) 窓際({ 7 8 9 }) の({ }) 席({ 6 }) を({ }) 御({ }) 願い({ 3 4 }) し({ 3 4 })
ます({ }) 。({ 10 })
# Sentence pair (9) source length 8 target length 7
this is my first time diving .
NULL ({ }) 海({ }) に({ }) 潜る({ 6 }) の({ }) は({ }) 初めて({ 1 2 3 4 5 }) です({ }) 。({ 7 })
# Sentence pair (12) source length 8 target length 8
go straight until you see a drugstore .
NULL ({ 3 4 6 }) まっすぐ({ 2 }) 行く({ 1 } ) と({ }) 薬局({ 7 }) が({ }) 見え({ 5 }) ます({ })
。({ 8 })
# Sentence pair (21) source length 5 target length 6
pass the bread , please .
NULL ({ 2 }) パン({ 3 }) を({ }) 回し({ 1 }) て下さい({ 4 5 }) 。({ 6 })
```

Table 3.2: IWSLT hand-annotated corpus. Note that although we use the GIZA++ format (which is called a A3 final file) this is not the result of word alignment, but the hand annotation itself. We employ this format to simplify the annotation of alignment links.

[3]The guidelines we adopt include:

- Unless there is no ‘of (EN)’, we do not align particles such as ‘no (JP)’. Hence, most particles are not aligned.
- We allow links between a punctuation mark and a word.
- Most proverbs in EN are not aligned since there is no correspondence in JP (Omission of subjects in JP).
- Most articles in EN are not aligned (No articles in JP).
- Although Japanese may include expressions which are related to the politeness register, we avoid the alignment of such words.
- We align ‘to (EN)’ combined with a verb to their corresponding Japanese verbs or nouns.

We established such rules not because we think these rules are the best, but because we need to establish consistent rules in order to label in a consistent manner. This is because there are two many options to label them since we assume that Japanese side is morphologically segmented.

3.3 Evaluation

We use BLEU (Papineni et al., 2002) as the extrinsic evaluation measure and AER (Och and Ney, 2003) as our intrinsic evaluation measure. Let c be the length of the testset and r be the length of the reference translation. For given precision p_n of n-grams of size up to N, BLEU is defined as in (3.1):

$$\text{BLEU-4} \quad = \quad \text{BP} \cdot \exp\left(\sum_{n=1}^{4} \log p_n\right) \tag{3.1}$$

where brevity penalty (BP) is intended to reduce the score if the output is too short as in (3.2):

$$\text{BP} \quad = \quad \min\left(1, e^{1-\frac{r}{c}}\right) \tag{3.2}$$

Alignment error rate (Och and Ney, 2003) is defined via the set of sure alignments S, possible alignments P, and whole alignments A. Recall is defined on S while precision is defined on P where $P \supset S$. These definitions are shown as in (3.3):

$$\begin{cases} \text{Precision}(A,P) &= \dfrac{|A \bigcap P|}{|A|}, \\ \text{Recall}(A,S) &= \dfrac{|A \bigcap S|}{|S|}, \\ \text{AER}(A,P,S) &= 1 - \dfrac{|A \bigcap S| + |A \bigcap P|}{|A| + |S|} \end{cases} \tag{3.3}$$

First, note that if the performance of a word alignment system is better than some other one, both recall and precision often increase. The normal experience in Machine Learning is that if recall goes up precision should go down, or vice versa. For this reason, this may look odd at first sight. However, this comes from the difference above that recall and precision are defined on slightly different objects. Second, note that it is not easy to calculate false negatives since we do not know how many alignment links existed for a pair of sentences. Hence, it is not easy to write ROC

(Receiver-Operating Characteristic; The ROC curve was used by the US army during World War II for the analysis of radar signals (Green and Swets, 1966)). One way to calculate this would be take the maximum between the number of source and target words.

Figure 3.1 shows the performance of GIZA++ on EN-FR Hansard datasets whose training set is 1.1 million sentence pairs, consisting of 10 iterations of IBM Model 1, 10 iterations of HMM Model, 10 iterations of IBM Model 3 and 10 iterations of Model 4. Figure 3.2 shows the

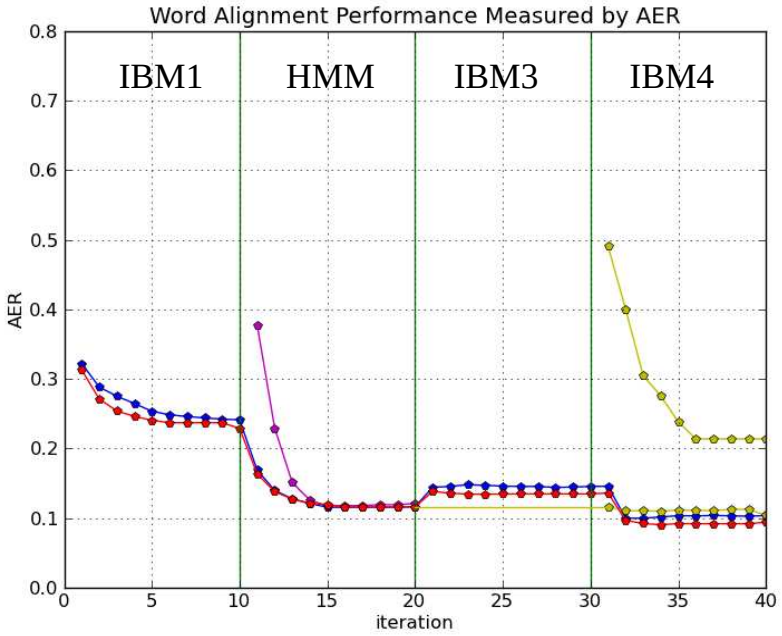

Figure 3.1: AER performance on EN-FR. The line which starts from the 11th iteration shows that this is only trained by the HMM model. The other line which starts from the 31st iteration shows that this is only trained by IBM Model 4. The red and blue lines show that they are different translation directions. Training set was Hansard 1.1 million sentence pair together with a hand-annotated amount of 484 sentence pairs.

performance of GIZA++ on the EN-JP corpus. The first observation is that the performance on EN-JP is considerably worse than the performance on EN-FR. In the case of EN-FR, it achieves 0.11 AER between 15 to 20 iterations and 0.09 AER between 34 to 40 iterations. In the case of EN-JP, it achieves 0.52 on IWSLT and 0.62 on NTCIR. The second observation is the variability

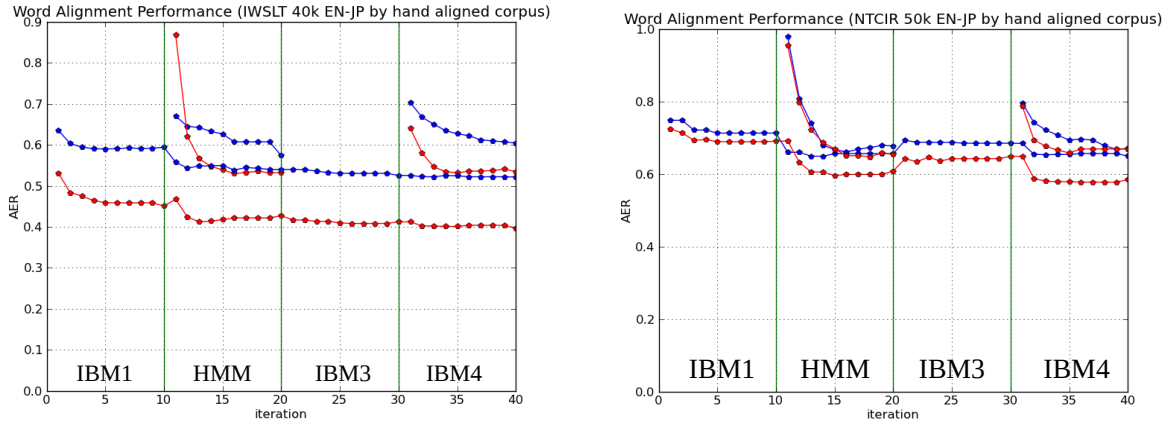

Figure 3.2: AER performance on EN-JP. The left figure shows the performance on the IWSLT corpus, and the right figure shows the performance on the NTCIR corpus. Training set was IWSLT 40k sentence pairs including a hand-annotated portion, while that of NTCIR was NTCIR 200k sentence pairs including a hand-annotated portion.

for different translation directions. In the case of EN-FR, the red and blue lines are quite identical, while in the case of EN-JP, the red line is always better than the blue line.

3.4 Bayesian Learning

In Machine Learning there are two ways of performing estimation: single point estimate and multiple point estimate. The former is taken in frequentist methods, such as Maximum Likelihood (ML) / Maximum-A-Posteriori (MAP) estimation, as well as word alignment using IBM Models. Multi-point estimation is used in a small amount of Bayesian methods (alternatively called a *full Bayesian method*).

Under the existence of latent variables, the *full Bayesian method* is not to consider a single point estimate of parameter θ, but rather evaluate it at many different θs. In order to consider the merit of this, let us consider a fair bet casino problem. This problem is to detect whether the coin is biased or fair from the observation, say 10 coin flips.

Let t be latent variables, w be observed variables, θ be the probability of heads (parameters),

n_H be the number of heads in w, and n_T be the number of tails in w. We are given a biased coin (t=1) with probability 0.5, or a fair coin (t=0) with probability 0.5. When the coin is biased, we assume a uniform distribution over θ, otherwise $\theta = 0.5$. Assume that we have a uniform prior on θ with $P(\theta) = 1$ for all $\theta \in [0, 1]$.

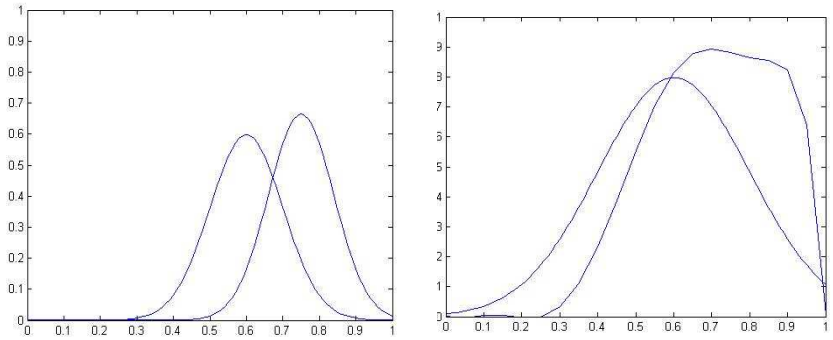

Figure 3.3: The left figure shows the simple point estimate and the right figure shows the multiple point estimate.

The single point estimate approach to this problem is conducted in two stages. The first step is to apply the MAP estimate for θ, as in (3.4):

$$\hat{\theta} \quad = \quad \arg\max_{\theta} P(w|\theta)P(\theta) \qquad \text{MAP estimate} \tag{3.4}$$

Now we observed a head n_H times, so the MAP estimate is $\hat{\theta} = n_H/10$. The second step is to look at the value of t that maximizes $P(t|w, \hat{\theta})$.

The multiple point estimate approach (or the full Bayesian approach) is to see the distribution over latent variables given the observed data, as in (3.5):

$$P(t|w) \quad = \quad \int P(t|w, \theta)P(\theta|w)d\theta \tag{3.5}$$

Using the case when $w = HHTHTTHHTH$ and $w = HHTHHHTHHH$ we can draw the

situation in Figure 3.3. The left figure shows the case of the single point estimation, while the right figure shows the case of the multiple point estimation. In the case of $w = HHTHTTHHTH$, both methods will obtain the best value of $\theta = 0.8$. In the case of $w = HHTHHHTHHH$, the multiple point estimation obtains a flat plateau around $\theta = 0.6 \sim 0.85$, which contrasts with the single point estimation of $\theta = 0.8$.

3.5 Algorithmic Design (I): Learning Generative Models (Exact Inference)

In this section, we deal with algorithms considering the case where there are no Markov dependencies, which is equivalent to the IBM Model 1. We first show the word alignment implemented by EM with the maximum likelihood estimates. We mention one characteristic of word alignment in that only a small amount of parameters (a matrix of t-table t) are updated at a time on a sentence by sentence basis.

Then, we move to consider how to incorporate prior knowledge into the formula. We apply the prior in the same EM model by replacing the M-step of maximum likelihood with MAP estimates. We show two ways to set up priors: cache-based or global-based. Note that MAP-EM has appeared in some papers (Bishop, 2006), albeit without much application as far as we know.

However, it turned out that this approach has one disadvantage in its MAP estimation in that it is inherently basis dependent (Beal, 2003; Koller and Friedman, 2009). Given θ^* has a non-zero prior probability, it is always possible to find a basis in which any particular θ^* is the MAP solution. This is the motivation of the third algorithm, which we call VB-EM algorithm (variational Bayesian EM algorithm), which is not basis-dependent. The VB-EM algorithm discusses the lower bound of the MAP-EM algorithm, modifying the distribution with a simpler form.

Note that the aim of word alignment is to find the most probable alignment path in the form of A3 final files, which is derived via Viterbi decoding which we describe Section 3.6.1. Practically, just one iteration of the HMM model is employed [4] if we need to derive such a path in IBM Model

[4]This is a technique which is often used when we want to obtain the Viterbi alignment or the A3 final file.

1.

3.5.1 Standard EM (standard ML, GIZA++)

An EM algorithm was formulated by Dempster et al. (Dempster et al., 1977), this algorithm has been implicitly used in various fields (Dempster et al., 1977; McLachlan and Krishnan, 1997; Bishop, 2006). First we introduce this algorithm, then we apply this to word alignment (Brown et al., 1993; Koehn, 2010).

Let Y be the random vector corresponding to the observed data y, having probability density function $g(y;\Phi)$ where $\Phi = (\Phi_1, \ldots, \Phi_d)^T$ is a vector of unknown parameters with parameter space Ω. The observed data vector y is viewed as being incomplete and is regarded as an observable function of the complete-data. Note that the incomplete data includes the missing data.

Let X be the random vector corresponding to the complete-data vector x, having probability density function $g_c(x;\Phi)$. The complete log-likelihood function is given by (3.6):

$$\log L_c(\Phi) \quad = \quad \log g_c(x;\Phi). \tag{3.6}$$

The relation between x and y is as in (3.7):

$$g(y;\Phi) \quad = \quad \int_{\mathbf{X}(y)} g_c(x;\Phi)dx \tag{3.7}$$

where $\mathbf{X}(y)$ is the subset of $\mathbf{X}$ in the relations of $y = y(x)$.

Let $\Phi^{(0)}$ be initial value for Φ. In the first iteration of E-step requires to calculate the following quantity $Q(\Phi;\Phi^{(0)})$, as in (3.8):

$$Q(\Phi;\Phi^{(0)}) = \mathbb{E}_{\Phi^{(0)}}\left(\log L_c(\Phi)|y\right). \tag{3.8}$$

The M-step is to maximize $Q(\Phi;\Phi^{(0)})$ with regard to Φ over the parameter space Ω. Hence, we

choose $\Phi^{(1)}$) such that $Q(\Phi^{(1)};\Phi^{(0)}) \geq Q(\Phi;\Phi^{(0)})$ for all $\Phi \in \Omega$. This is equivalent to (3.9):

$$M(\Phi^{(1)}) \quad = \quad \arg\max_{\Phi} Q(\Phi;\Phi^{(0)}). \tag{3.9}$$

Then this iteration is repeated as follows.

- E-step: Calculate $Q(\Phi;\Phi^{(k)})$ where $Q(\Phi;\Phi^{(k)}) = \mathbb{E}_{\Phi^{(k)}}\left(\log L_c(\Phi)|y\right)$.
- M-step: Choose $\Phi^{(k+1)}$ which maximizes $Q(\Phi;\Phi^{(k)})$. That is $Q(\Phi^{(k+1)};\Phi^{(k)}) \geq Q(\Phi;\Phi^{(k)})$ for all $\Phi \in \Omega$, or the alternative equivalent representation in (3.10):

$$M(\Phi^{(k)}) \quad = \quad \arg\max_{\Phi} Q(\Phi;\Phi^{(k)}). \tag{3.10}$$

Dempster et al. (Dempster et al., 1977) show that the incomplete-data likelihood function $L(\Phi)$ is not decreased after an EM iteration, as in (3.11):

$$L(\Phi^{(k+1)}) \geq L(\Phi^{(k)}) \tag{3.11}$$

for $k = 0, 1, 2, \ldots$. Hence, the likelihood values are bounded above and convergence is achieved.

In IBM Model 1, $P(f, a|e)$ is defined as in (3.12):

$$P(f, a|e) \quad = \quad \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} t(f_j|e_{a_j}) \tag{3.12}$$

Since we do not know the alignment function a we do not know $P(f, a|e)$ directly. However, we can calculate the expectation of $P(f, a|e)$, that is $\sum P(f, a|e)$ (or $\mathbb{E}P(f, a|e)$, the Bayesian average of $P(f, a|e)$) enumerating all the possible alignment functions a. Suppose we are given such expectation $\sum P(f, a|e)$ (which is equivalent to $P(f|e)$), we can obtain $P(f, a|e)$ by maximizing the this expectation $\sum P(f, a|e)$ with respect to $P(f|e)$, which is called the Expectation

Maximization algorithm (Dempster et al., 1977). Now we have the equality constraint for each e, M-step can be written as in (3.14):[5]

$$\begin{cases} \text{maximize} & \sum t(f,a|e)\,(=t(f|e)) & = & \dfrac{\epsilon}{(l+1)^m}\displaystyle\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\prod_{j=1}^{m}t(f_j|e_{a_j}) \\ \text{subject to} & \sum_f t(f|e) & = & 1 \end{cases} \tag{3.14}$$

Note that our training corpus consists of a set of translations, $(e^{(1)}|f^{(1)})$, $(e^{(2)}|f^{(2)})$,...,$(e^{(S)}|f^{(S)})$. This fact corresponds to the fact that the alignment function is closed within each sentence pair, i.e. a constraint on the range of a map a_i.

In order to find the maxima (or minima) with equality constraint of (3.14), we use the Lagrange method of Lagrange (1797; general introduction is available in e.g. (Cristianini and Shawe-Taylor, 2000; Bishop, 2006)). Let λ_e denotes a Lagrange multiplier. Lagrangian $\mathcal{L}(t,\lambda)$ can be written as in (3.15):

$$\mathcal{L}(t,\lambda) = \frac{\epsilon}{(l+1)^m}\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\prod_{j=1}^{m}t(f_j|e_{a_j}) - \sum_e \lambda_e\left(\sum_f t(f|e) - 1\right) \tag{3.15}$$

Then, the partial derivative of $\mathcal{L}(t,\lambda)$ with respect to t becomes as in (3.37):

$$\frac{\partial\mathcal{L}(t,\lambda)}{\partial t} = \frac{\epsilon}{(l+1)^m}\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\sum_{j=1}^{m}\delta(f,f_j)\delta(e,e_{a_j})t(f|e)^{-1}\prod_{k=1}^{m}t(f_k|e_{a_k}) - \lambda_e \tag{3.16}$$

where δ is the Kronecker delta function, equal to 1 when both of its arguments are the same and 0

[5]It is noted that (3.14) is equivalent to (3.13):

$$\begin{cases} \text{maximize} & \mathbb{E}t(f,a|e)\,(=t(f|e)) & = & \dfrac{\epsilon}{(l+1)^m}\displaystyle\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\prod_{j=1}^{m}t(f_j|e_{a_j}) \\ \text{subject to} & \sum_f t(f|e) & = & 1 \end{cases} \tag{3.13}$$

otherwise. The stationary point is attained when this partial derivative is zero as in (3.17):

$$t(f|e) = \lambda_e{}^{-1}\frac{\epsilon}{(l+1)^m}\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\sum_{j=1}^{m}\delta(f,f_j)\delta(e,e_{a_j})\prod_{k=1}^{m}t(f_k|e_{a_k}) \quad (3.17)$$

$$= \lambda_e{}^{-1}\frac{\epsilon}{(l+1)^m}\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\prod_{k=1}^{m}t(f_k|e_{a_k})\{\sum_{j=1}^{m}\delta(f,f_j)\delta(e,e_{a_j})\} \quad (3.18)$$

$$(3.19)$$

$$= \lambda_e{}^{-1}\frac{\epsilon}{(l+1)^m}\prod_{k=1}^{m}\sum_{i=0}^{l}t(f_k|e_i)\{\sum_{j=1}^{m}\delta(f,f_j)\delta(e,e_{a_j})\} \quad (3.20)$$

This suggests that given an initial guess for the translation probabilities $t(f_k|e_{a_k})$ in the righthand side of (3.17), we will obtain a new estimate of $t(f|e)$ in the lefthand side. Note that we use (3.21) to change the order of $\sum$ and $\prod$ from (3.19) to (3.20).

$$\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\prod_{k=1}^{m}t(f_k|e_{a_k}) = \prod_{k=1}^{m}\sum_{i=0}^{l}t(f_k|e_i) \quad (3.21)$$

In order to facilitate the computation, we define an auxiliary function $c(f|e)$ to remove ϵ.

$$c(f|e) = \sum_a p(a|e,f)\sum_{j=1}^{l_e}\delta(e,e_j)\delta(f,f_{a(j)}) \quad (3.22)$$

$$= \frac{t(e|f)}{\sum_{i=0}^{l_f}t(e|f_i)}\sum_{j=1}^{l_e}\delta(e,e_j)\sum_{i=0}^{l_f}\delta(f,f_i) \quad (3.23)$$

This function $c(f|e)$ is calculated for each sentence pair, from the sentence pair $(e^{(1)},f^{(1)})$ to $(e^{(S)},f^{(S)})$. Then, we sum all of these to obtain $\sum_S c(f|e)$.

If we first compute *total_s[e]*$=\sum_{i=0}^{l_f}t(e|f_i)$, then we can calculate *count(e|f)*$= t(e|f)\sum_{j=1}^{l_e}\delta(e,e_j)\sum_{i=0}^{l_f}\delta(f,f_i)/$ *total_s[e].* The number of operations to calculate becomes proportional to $l+m$ rather than to $(l+1)^m$ which is suggested by $c(f|e)=\sum_a P(a|e,f)\sum_{j=1}^{m}\delta(f,f_j)\delta(e,e_{a_j})$. Note that the calculation is done in the training sentence pair basis. For the u-th sentence pair, the

cached auxiliary function $c(f|e, f^{(u)}, e^{(u)})$ is calculated. Then, $c(f|e)$ is obtained by calculating $\sum_{1 \leq u \leq S} c(f|e, f^{(u)}, e^{(u)})$.

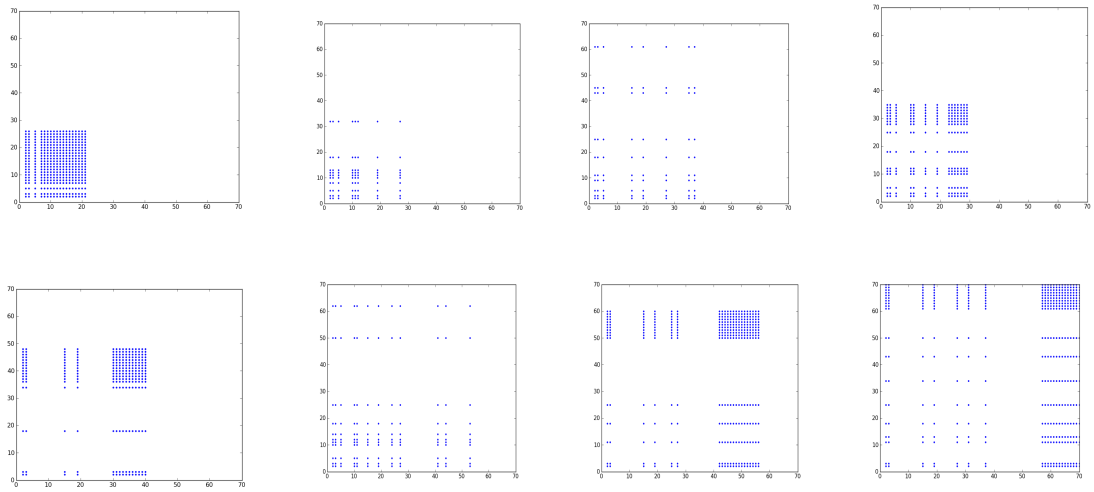

Figure 3.4: A full matrix shows an example of the entire words in English (x-axis) and French (y-axis). Eight figures show each iteration of examined word pairs by the word alignment process. Notice that there are many word (cept) pairs which are not examined (Refer to Figure 3.5).

Then, what we need is to calculate

$$t(e|f) = \frac{(\sum_S c(e|f))}{\sum_e (\sum_S c(e|f))} \tag{3.24}$$

$$= \frac{\text{count(e|f)}}{\textit{total\_s[f]}} \tag{3.25}$$

As is evident with this definition of total_s[f] $= \lambda_e$, total_s[f] works as the normalization constant.

Listing 3.1 shows the algorithm of word alignment (IBM Model 1), which aims at obtaining the t-table shown in an array t. One characteristic of word alignment relates to the line 21. Given the training corpus, each line is processed line by line updating a small portion of t-table t. Figure 3.4 shows this situation in eight iterations, starting from the upper left figure to the bottom right figure. The x-axis shows the word ID of English word e and the y-axis shows the word ID of French word f. We marked whole the points representing a word pair e and f where the alignment between this

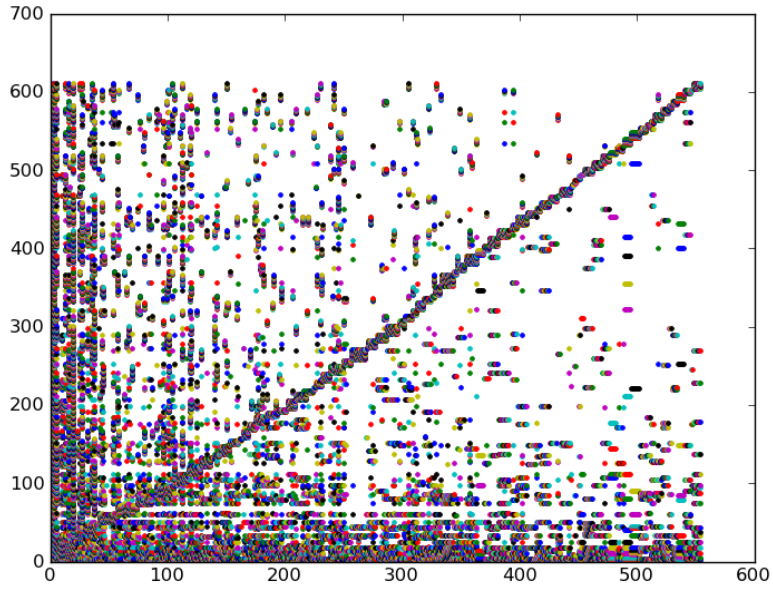

Figure 3.5: A matrix shows that an example of word pairs which are examined by word alignment process are plotted (black), while word pairs not examined are not plotted (white). The corpus is EN-FR hansard 488 sentences.

word pair is examined in the word alignment process, while we do not mark the points where the alignment will not be examined. Then, we noticed that there are a considerable number of word pairs which are not considered at all from the beginning. This is shown in Figure 3.4, where the points which are not marked are the points which are not considered. One way to utilize this in the algorithm is to make a sparse matrix t.

Listing 3.1: wordAlignment.py

```
1 def trainStandardEM(L1tok, L2tok):
2     t = {}
3     count = {}
4     total = {}
5     total_s = {}
6     residual=1
7     print 'Initial correspondence calculation'
8     numSentence = len(L1tok)
9     assert(len(L1tok)==len(L2tok))
```

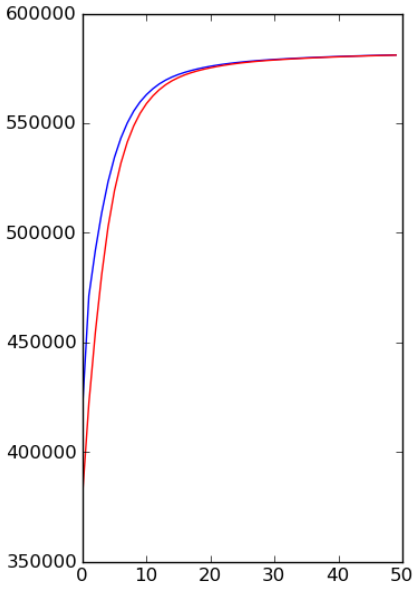

Figure 3.6: Convergence in terms of log likelihood.

```
    iteration = 0
    print 'Initial parameters set.'
    for i in range(numSentence):
        for e in L1tok[i]:
            for f in L2tok[i]:
                t[(e,f)] = UNIFORM_PROB
    print 'Iterations.'
    PPs = []
    while (iteration < ITERATION_MAX):
        count = {}
        total = {}
        for i in range(numSentence):
            for e in L1tok[i]:
                total_s[e] = 0
                for f in L2tok[i]:
                    try:
                        total_s[e] += t[(e,f)]
                    except:
                        pass
            for e in L1tok[i]:
                for f in L2tok[i]:
                    try:
```

```
                        count[(e,f)] += t[(e,f)] / total_s[e]
                    except KeyError:
                        try:
                            count[(e,f)] = t[(e,f)] / total_s[e]
                        except:
                            pass
                    try:
                        total[f] += t[(e,f)] / total_s[e]
                    except KeyError:
                        try:
                            total[f] = t[(e,f)] / total_s[e]
                        except:
                            pass
        for x in count.keys():
            f=x[1]
            try:
                t[x] = count[x] / total[f]
            except KeyError:
                pass
        print
        perplexity = - np.sum(np.log2(x[1]) for x in t.items())
        print 'PP=',perplexity
        PPs.append(perplexity)
    return t
```

Figure 3.6 shows the convergence in terms of log likelihood. Although we often iterate 5 iterations or 10 iterations on IBM Model 1 when we use GIZA++, this graph shows that the convergence is achieved around 20 iterations.

3.5.2 Standard EM (standard ML, Vectorial Form)

Now we can easily modify this algorithm in vectorial form, which takes advantages if this is run on some architecture of CPUs or software such as Matlab. Pseudocode is shown in Listings A.1 in Appendix.

3.5.3 Local MAP Estimate-EM (standard ML)

Let $P(a_i)$ be prior knowledge about a local alignment link from the position i in f to the position a_i in e. $P(f, a|e)$ for MAP estimate version of IBM Model 1 can be defined as in (3.26):

$$P(f, a|e) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} P(f_j|e_{a_j}, a_j)P(a_j) \tag{3.26}$$

Then,

$$\begin{cases} \text{maximize} & \sum P(f, a|e) = \dfrac{\epsilon}{(l+1)^m} \displaystyle\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} P(f_j|e_{a_j}, a_j)P(a_j) \\ \text{subject to} & \sum_f t(f|e) = 1 \end{cases} \tag{3.27}$$

The Lagrangian $\mathcal{L}(t, \lambda)$ can be written as in (3.36):

$$\mathcal{L}(t, \lambda) = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} P(f_j|e_{a_j}, a_j)P(a_j) - \sum_e \lambda_e \left(\sum_f t(f|e) - 1 \right) \tag{3.28}$$

Then, the partial derivative of $\mathcal{L}(t, \lambda)$ with respect to t becomes as in (3.37):

$$\frac{\partial \mathcal{L}(t, \lambda)}{\partial t} = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j})t(f|e)^{-1} \tag{3.29}$$

$$\prod_{k=1}^{m} t(f_k|e_{a_k}, a_k)P(a_k) - \lambda_e \tag{3.30}$$

where δ is the Kronecker delta function. The stationary point is attained when this partial derivative is zero as in (3.31):

$$\begin{aligned} t(f|e) &= {\lambda_e}^{-1}\frac{\epsilon}{(l+1)^m}\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\sum_{j=1}^{m}\delta(f,f_j)\delta(e,e_{a_j})\prod_{k=1}^{n}P(f_k|e_{a_k},a_k)P(a_k) \quad (3.31)\\ &= {\lambda_e}^{-1}\frac{\epsilon}{(l+1)^m}\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\prod_{k=1}^{n}P(f_k|e_{a_k},a_k)P(a_k)\{\sum_{j=1}^{m}\delta(f,f_j)\delta(e,e_{a_j})\}\\ &= {\lambda_e}^{-1}\frac{\epsilon}{(l+1)^m}\prod_{k=1}^{m}\sum_{i=0}^{l}t(f_k|e_i)P(a_k) \end{aligned}$$

This suggests that given an initial guess for the translation probabilities $t(f_k|e_{a_k})$ on the righthand side of (3.31), we will obtain a new estimate of $t(f|e)$ on the lefthand side.

In order to facilitate the computation, we define an auxiliary function $c(f|e)$ to remove ϵ.

$$c(f|e) = \sum_{j=1}^{m}\sum_{i=0}^{n}\frac{t(e|f)P(a_k)\delta(e,e_j)\delta(f,f_i)}{\sum_{i'=0}^{l_f}t(e|f_{i'})P(a_{i'})} \quad (3.32)$$

The prior $\log p(t)$, a probability used to reflect the degree of prior belief about the occurrences of the events, can embed prior knowledge about noun phrases (or MWEs). The prior of this MAP-based word aligner is defined by the alignment links between e and f by $T = \{(sentID, t_i, t_j, pos_i, pos_j), \ldots,\}$. We use this information to calculate the prior $p(t) = p(t; e, f, T)$ for the given words e and f: this is 1 if e and f have an alignment link, 0 if they are not connected, and uniform if their link is not known. This is shown in (3.33):[6]

$$p(t; e_i, f_i, T) = \begin{cases} 1 & (e_i = t_i, f_j = t_j)\\ 0 & (e_i = t_i, f_j \neq t_j)\\ 0 & (e_i \neq t_i, f_j = t_j)\\ \text{uniform} & (e_i \neq t_i, f_j \neq t_j) \end{cases} \quad (3.33)$$

[6]This equation shows how to set up our prior. For example, when $e_i = t_i and f_j \neq t_j$, our prior takes the value 0. This is the value of the prior.)

pair	GIZA++(no prior)			Ours(with prior)		
EN-FR	fin	ini	prior	fin	ini	prior
is *NULL*	1	.25	0	0	.25	.25
rosy *en*	1	.5	0	0	.5	.2
that .	1	.25	0	0	.25	.25
life *la*	1	.25	0	0	.25	0
. *c‘*	1	.25	0	0	.25	.25
that *c‘*	0	.25	0	1	.25	.25
is *est*	0	.25	0	1	.25	.25
life *vie*	0	.5	0	1	.5	1
rosy *rose*	0	.25	0	1	.25	.2

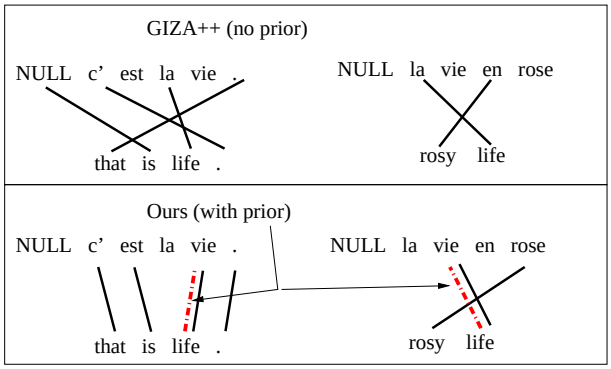

Table 3.3: Benefit of prior knowledge about anchor words illustrated by toy data.

The posterior probability is calculated for the prior as in the Listing 3.3.

Listing 3.2: wordAlignment.py

```
def trainMAPEM(L1tok,L2tok,prior):
    ...the same as trainStandardEM ...
    while (iteration < ITERATION_MAX):
        count = {}
        total = {}
        for i in range(numSentence):
            for e in L1tok[i]:
                total_s[e] = 0
                for f in L2tok[i]:
                    try:
                        total_s[e] += t[(e,f)] * pri[(e,f)]
                    except:
                        pass
            for e in L1tok[i]:
```

```
                for f in L2tok[i]:
                    try:
                        count[(e,f)] += t[(e,f)] * pri[(e,f)] /
total_s[e]
                    except KeyError:
                        try:
                            count[(e,f)] = t[(e,f)] * pri[(e,f)] /
total_s[e]
                        except:
                            pass
                    try:
                        total[f] += t[(e,f)] * pri[(e,f)] / total_s[e]
                    except KeyError:
                        try:
                            total[f] = t[(e,f)] * pri[(e,f)] /
total_s[e]
                        except:
                            pass
        for x in count.keys():
            f=x[1]
            try:
                pri=prior[f]
            except:
                pri=1.0
            try:
                t[x] = count[x] / total[f]
            except KeyError:
                pass
        logLikelihood = -sum(log(x[1]) for x in t.items())
        for x in count.keys():
            f=x[1]
            if (t[x] < PROB_SMOOTH):
                t[x]=PROB_SMOOTH
    ...the same as trainStandardEM ...
```

3.5.4 MAP Assignment-EM (standard ML)

Except for the simplest case when we do not consider any Markov dependencies, the problem of MAP estimate corresponds to finding the configuration that is most likely under the distribution $p(x)$ defined over the (Bayesian or Markov) network. In this general setting, under the assumption that we do not doubt the evidences, the task of MAP assignment is to find the most likely

assignment to all of the (non-evidence) variables. This setting is somewhat simpler than the MAP estimate in the previous section in which we assume the prior belief which we may override depending on the statistics.

First we devide variables into the variables whose value are already known a priori and those which we would like to obtain. Concretely, let $j \in PRI$ be variables which we know the values a priori, and $j \in VAR$ be variables in question. Then, $P(f, a|e)$ for the MAP assignment version of IBM Model 1 can be defined as in (3.34):

$$P(f,a|e) = \frac{\epsilon}{(l+1)^m} \prod_{j \in VAR}^{m} t(f_j|e_{a_j}) \prod_{j \in PRI} t(f_j|e_{a_j}) \tag{3.34}$$

Then,

$$\begin{cases} \text{maximize} & \sum_a t(f,a|e) = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j \in VAR} t(f_j|e_{a_j}) \\ & \qquad\qquad \prod_{j \in PRI} t(f_j|e_{a_j}) \\ \text{subject to} & \sum_f t(f|e) = 1 \end{cases} \tag{3.35}$$

Then, the Lagrangian $\mathcal{L}(t, \lambda)$ can be written as in (3.36):

$$\mathcal{L}(t,\lambda) = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j \in VAR} t(f_j|e_{a_j}) \prod_{j \in PRI} t(f_j|e_{a_j}) - \sum_e \lambda_e \left(\sum_f t(f|e) - 1 \right) \tag{3.36}$$

Then, the partial derivative of $\mathcal{L}(t, \lambda)$ with respect to t becomes as in (3.37):

$$\frac{\partial \mathcal{L}(t,\lambda)}{\partial t} = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j}) t(f|e)^{-1} \prod_{k \in VAR} t(f_k|e_{a_k}) \prod_{k \in PRI} t(f_k|e_{a_k}) - \lambda_e \tag{3.37}$$

where δ is the Kronecker delta function. The stationary point is attained when this partial derivative is zero as in (3.38):

$$
\begin{aligned}
& t(f|e) \\
&= \lambda_e{}^{-1} \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j}) \prod_{k \in VAR} t(f_k|e_{a_k}) \prod_{k \in PRI} t(f_k|e_{a_k}) \\
&= \lambda_e{}^{-1} \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{k \in VAR} t(f_k|e_{a_k}) \prod_{k \in PRI} t(f_k|e_{a_k}) \{\sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j})\} \\
&= \lambda_e{}^{-1} \frac{\epsilon}{(l+1)^m} \sum_{a_{v0} \in VAR} \cdots \sum_{a_{vn} \in VAR} \prod_{k \in VAR} t(f_k|e_{a_k}) \sum_{a_{p0} \in PRI} \cdots \sum_{a_{pm} \in PRI} \\
&\quad \prod_{k \in PRI} t(f_k|e_{a_k}) \{\sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j})\}
\end{aligned}
\tag{3.38}
$$

This suggests that given an initial guess for the translation probabilities $t(f_k|e_{a_k})$ on the righthand side of (3.38), we will obtain a new estimate of $t(f|e)$ on the lefthand side.

In order to facilitate the computation, we define an auxiliary function $c(f|e)$ to remove ϵ.

$$
\begin{aligned}
c(f|e) &= \sum_a p(a|e, f) \sum_{j=1}^{l_e} \delta(e, e_j)\delta(f, f_{a(j)}) \\
&= \frac{t(e|f)}{\sum_{i \in VAR}^{l_f} t(e|f_i) + \sum_{i \in PRI} t(e|f_i)} \sum_{j=1}^{l_e} \delta(e, e_j) \sum_{i=0}^{l_f} \delta(f, f_i)
\end{aligned}
\tag{3.39}
$$

3.5.5 Variational Bayesian-EM (standard ML)

The Bayesian Machine Learning algorithm often depends on the assumed complexity of the distribution. A simple algorithm, such as the maximum entropy maximization algorithm, only assumes the exponential family distribution. An exponential family is the set of parametric distributions that has sufficient statistics and a consistent maximum likelihood estimate. In other words, the likelihood cost function has a global optimum and the estimate of the parameters of distributions

in this family will be unique,[7] as in (3.40):

$$P(X|\theta) \quad = \quad \exp(A(X) + T(X)^\top \theta - K(\theta)). \tag{3.40}$$

where $A(X)$ denotes a function of the data, $K(\theta)$ a function of the model or the partition function, and $T(X)^\top \theta$ an inner product between the model and a function of the data. The exponential family distribution includes most of the naturally arising distributions, such as normal, binomial, Dirichlet, Laplace, Wishart distributions, and so forth. The partition function $K(\theta)$ ensures that the distribution is normalized when we integrate over X. More complex distributions are taken in the form of mixture models. Further complicated distributions are described by graphical models. The standard EM and the MAP-EM which we explained in the last two sections are tools to handle mixture models, and HMM models in general which we explain in the next chapter fall into the category of graphical models. The complexity of distribution depends on the nature of the application.

This section follows the description of Beal (Beal, 2003). The idea of the variational Bayesian framework employs a simpler distribution than a true distribution to approximate the distribution over both hidden variables and parameters. Often, we assume a simpler distribution, which we call the *free distribution* $q(x, \theta)$, where the hidden states and parameters are independent given the data.

We assume a prior distribution over parameters $p(\theta|m)$ conditioned on the model m. The marginal likelihood of a model $p(y|m)$ can be lower bounded by introducing any distribution over both latent variables and parameters which has support $p(x, \theta|y, m)$.

Let m be a model with parameters θ giving rise to an IID data set $y = \{y_1, \ldots, y_n\}$ with corresponding hidden variables $x = \{x_1, \ldots, x_n\}$. A lower bound on the model, which is the

[7] Exponential families are characterized by their strictly convex and differential functions K, which is called a log-normalizer (or a partition function). Due to such properties, the Hessian of this log-normalizer is a positive definite matrix.

model log marginal likelihood, is as in (3.41):

$$F_m(q_x(x), q_\theta(\theta)) \quad = \quad \int d\theta dq_x(x) q_\theta(\theta) \log \frac{p(x, y, \theta|m)}{q_x(x) q_\theta(\theta)}. \tag{3.41}$$

This can be iteratively optimised by performing the following updates, which will converge to a local maximum of $F_m(q_x(x), q_\theta(\theta))$. Note that the superscript t denotes the iteration number.

E-step:

$$q_{x_i}^{(t+1)}(x_i) \quad = \quad \frac{1}{Z_{x_i}} \exp\left[\int d\theta q_\theta^{(t)} \log p(x_i, y_i|\theta, m)\right] \qquad \text{for all i} \tag{3.42}$$

where

$$q_x^{(t+1)}(x) \quad = \quad \prod_{i=1}^{n} q_{x_i}^{(t+1)}(x_i). \tag{3.43}$$

M-step:

$$q_\theta^{(t+1)}(\theta) \quad = \quad \frac{1}{Z_\theta} \exp\left[\int dx q_x^{(t+1)}(x) \log p(x, y|\theta, m)\right] \tag{3.44}$$

Pseudocode is shown in Listing A.2 in Appendix. The difference from the standard EM is evident in line 31 where we calculate the lower bound in Equation (3.42) and the posterior probability in the M-step.

3.6 Algorithmic Design (II): Learning HMM

Although word alignment handles two word sequences in different languages, the EM models described in the last section do not take advantage of this: they do not treat them as proper *sequences*, but rather a *bag-of-words*. In this section, we try to treat them as sequences. Our usual setting is that we do not have a hand-annotated word aligned corpus; we do not have the (direct) labels, but

we can compute the expectation of such labels in terms of hidden variables, i.e. alignment functions. The HMM word alignment of Vogel et al. (Vogel et al., 1996) modeled alignment functions as hidden variables. In brief, the purpose of the HMM model is to introduce a first order Markov dependency in terms of states.

The discussion of HMM in this section resembles the discussion of the EM algorithm, but becomes substantially more complicated since the parameters θ in the model are not a single one but three, i.e. a translation matrix, an emission matrix, and an init matrix. The first and the second algorithms are the standard HMM algorithm. We show two implementations where the former uses the forward-backward algorithm (or Baum-Welch algorithm) (Baum et al., 1970) via EM algorithm, while the latter uses the graphical model. Since the standard HMM does not have the interface of prior knowledge, we incorporate priors on parameters, a transition matrix, an emission matrix, and an init matrix. This model is called a Bayesian HMM (Ghahramani, 2001; Stolcke, 2002). One algorithm counts the pseudo-counts given alternative sentences provided as a prior, which is the third algorithm which we call a pseudo-count Bayesian HMM, while another algorithm makes use of Gibbs sampling to estimate the E-step, which is the fourth algorithm which we call a Gibbs-Bayesian HMM. We have a potential difficulty in this Bayesian HMM in that our aim (which we discuss in detail in the next section) is to place the prior knowledge about alignment links. This difficult problem is often called the MAP configuration problem, which is one of the hottest topics in the Bayesian Machine Learning community. All Bayesian HMM algorithms, as far as we know, are aimed at solving the overfitting problem: these algorithms aim at learning the parameters of priors. In contrast, we aim at enforcing priors in the model: our algorithm aims at learning models under the specified priors which are already set. Similar to the discussion regarding the EM algorithm, the MAP estimation is basis-dependent. We demonstrate a variational Bayesian approach to HMM, which we call a *variational Bayesian-HMM* (VB-HMM).

3.6.1 Standard HMM (standard ML, GIZA++): Baum-Welch Implementation

HMM Model handles not a bag-of-word but a sequence. Hence, all the possible alignments are not equally likely possible. Instead of enumerating all the possible alignments $\sum_{a_1^m}$ as in (3.46), we can use the maximum approximation $\max_{a_1^m}$ as in (3.47) (Vogel et al., 1996; Ney et al., 2000):

$$
\begin{aligned}
P(\breve{f}|\breve{e}) &= \big(P(f_{1:l_f}|e_{1:l_e}) = P(f_1^l|e_1^m)\big) && (3.45)\\
&= \sum_{a_1^m}\prod_{j=1}^{J}\big[p(a_j|a_{j-1},I)p(f_j|e_{a_j})\big] && (3.46)\\
&\simeq \max_{a_1^m}\prod_{j=1}^{J}\big[p(a_j|a_{j-1},I)p(f_j|e_{a_j})\big] && (3.47)
\end{aligned}
$$

Let Z be a normalization constant. Recollect that in the IBM Model 1 $\sum_a t(f,a|e) = t(f|e)$. We can write (3.48):

$$
\begin{cases}
\text{maximize} \quad t(f|e) &= \dfrac{1}{Z}\max_{a_1^m}\prod_{j=1}^{m} t(f_j|e_{a_m},a_m)P(a_m|a_{m-1}) \\
\text{subject to} \quad \sum_f t(f|e) &= 1
\end{cases} \tag{3.48}
$$

Note that the standard HMM does not have an equality constraint in this formula. Similarly with the previous sections, we consider to employ the Lagrange method of Lagrange (1797). Let λ_e denotes a Lagrange multiplier. The Lagrangian $\mathcal{L}(t,\lambda)$ can be written as in (3.49):

$$
\mathcal{L}(t,\lambda) = \frac{1}{Z}\max_{a_1^l}\prod_{j=1}^{m} t(f_j|e_{a_m},a_m)P(a_m|a_{m-1}) - \sum_e \lambda_e\left(\sum_f t(f|e) - 1\right) \tag{3.49}
$$

The partial derivative of $\mathcal{L}(t,\lambda)$ with respect to t becomes as in (3.50):

$$
\frac{\partial\mathcal{L}(t,\lambda)}{\partial t} = \frac{1}{Z}\sum_{j=1}^{m}\delta(f,f_j)\delta(e,e_{a_j})t(f|e)^{-1}\max_{a_1^l}\prod_{k=1}^{m} t(f_k|e_{a_j},a_j)P(a_j|a_{j-1}) - \lambda_e \tag{3.50}
$$

where δ is the Kronecker delta function, equal to 1 when both of its arguments are the same and 0

otherwise. The stationary point is attained when this partial derivative is zero as in (3.51):

$$t(f|e) = {\lambda_e}^{-1}\frac{1}{Z}\sum_{j=1}^{m}\delta(f,f_j)\delta(e,e_{a_j})\max_{a_1^l}\prod_{k=1}^{m}t(f_k|e_{a_k},a_k)P(a_k|a_{k-1}) \tag{3.51}$$

We define an auxiliary function $c(f|e)$ to remove ϵ.

$$c(f|e) = \sum_a p(a|e,f)\sum_{j=1}^{l_e}\delta(e,e_j)\delta(f,f_{a(j)}) \tag{3.52}$$

$$\simeq \frac{t(e|f)}{\max_{a_1^l}\prod_{k=1}^{m}t(f_k|e_{a_k},a_k)P(a_k|a_{k-1})}\sum_{j=1}^{l_e}\delta(e,e_j)\sum_{i=0}^{l_f}\delta(f,f_i) \tag{3.53}$$

This function $c(f|e)$ is calculated for each sentence pair, from the sentence pair $(e^{(1)},f^{(1)})$ to $(e^{(S)},f^{(S)})$. The quantity $\max_{a_1^l}\prod_{k=1}^{m}t(f_k|e_{a_k},a_k)P(a_k|a_{k-1})$ can be obtained via the standard HMM where we use a cache matrix for each sentence pair. Then, we sum all of these to obtain $\sum_S c(f|e)$.

$$t(e|f) = \frac{(\sum_S c(e|f))}{\sum_e(\sum_S c(e|f))} \tag{3.54}$$

Now we focus on the standard HMM to obtain the quantity $\max_{a_1^l}\prod_{k=1}^{m}t(f_k|e_{a_k},a_k)P(a_k|a_{k-1})$.

The description of HMM model is based on Gharamani (2001). The HMM model assumes that the state of this hidden process satisfies the Markov property: given the value of S_{t-1}, the current state S_t is independent of all the states prior to $t-1$. Figure 3.7 shows a graphical model of an HMM. Let $X_{1:T}$ denote $X_1,\ldots,X_T$. We can write the complete-data likelihood of a sequence as in (3.55):

$$p(s_{1:T},y_{1:T}) = p(s_1)p(y_1|s_1)\prod_{t=2}^{T}p(s_t|s_{t-1})p(y_t|s_t) \tag{3.55}$$

where we marginalize out hidden variables $s_{1:T}$ as in (3.56):

$$p(y_{1:T}) = \sum_{s_{1:T}} p(s_{1:T}, y_{1:T}) \tag{3.56}$$

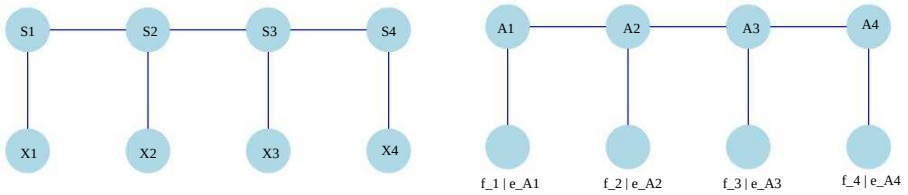

Figure 3.7: The left figure shows the original HMM model, while the right figure shows the HMM word alignment of Vogel et al. Note that the word pair appears at an emission node.

Let θ denote the set of parameters in the model. θ consists of a transition matrix A, an emission matrix C, and an init matrix π.

$$\theta = (A, C, \pi) \tag{3.57}$$

$$A = \{a_{jj'}\} : a_{jj'} = p(s_t = j' | s_{t-1} = j) \tag{3.58}$$

$$C = \{c_{jm}\} : c_{jm} = p(y_t = m | s_t = j) \tag{3.59}$$

$$\pi = \{\pi_j\} : \pi_j = p(s_1 = j) \tag{3.60}$$

We show the dependencies in $\theta = (A, C, \pi)$ in $p(s_t|s_{t-1})$, $p(y_t|s_t)$, and $p(s_1)$ as in (3.61) – (3.63), respectively.

$$p(s_t|s_{t-1}, A) = \prod_{j=1}^{k} \prod_{j'=1}^{k} a_{jj'}^{s_{t,j'} s_{t-1,j}} \tag{3.61}$$

$$p(y_t|s_t, C) = \prod_{j=1}^{k} \prod_{m=1}^{p} c_{jm}^{s_{t,j} y_{t,m}} \tag{3.62}$$

$$p(s_1|\pi) = \prod_{j=1}^{k} \pi_j^{s_{1,j}} \tag{3.63}$$

while we add the parameter dependencies in (3.55) as in (3.64) – (3.65):

$$p(s_{1:T}, y_{1:T}|\theta) = p(s_1|\pi)\prod_{t=1}^{T} p(y_t|s_t, C)\prod_{t=2}^{T} p(s_t|s_{t-1}, A) \tag{3.64}$$

$$= \log p(s_1|\pi) + \sum_{t=1}^{T} \log p(y_t|s_t, C)\sum_{t=2}^{T} \log p(s_t|s_{t-1}, A) \tag{3.65}$$

Note that depending on the case, the HMM can have discrete and continuous values. We explain only the case of discrete values since we do not need to consider the continuous case in word alignment.

Now, we consider the application of the EM algorithm since we cannot compute this in general as the state variables are hidden. We consider the expectation under the posterior distribution of the hidden states given the observations, as in (3.66):

$$\mathbb{E}f(x) = \int_{\mathcal{X}} f(X)P(X|Y, \theta)dX \tag{3.66}$$

and as in Listing A.3 in Appendix. This posterior distribution of the latent variables can be obtained efficiently using a two-stage message passing algorithm. In the HMM literature, this is known as the forward-backward algorithm (Rabiner, 1989), or the Baum-Welch algorithm (Baum et al., 1970). The forward-backward algorithm is one instance of the sum-product algorithm to obtain marginals in the graphical model.

Using the forward-backward algorithm in (3.70), we derive the following quantities.

$$A_{ij} = \sum_{t=2}^{T} \mathbb{E}s_{t,i}s_{t-1,j} \tag{3.67}$$

$$= \frac{\sum_{t=2}^{T} \mathbb{E}s_{t,i}s_{t-1,j}}{\sum_{t=2}^{T} \mathbb{E}s_{t-1,j}} \tag{3.68}$$

$$C_i = \mathbb{E}s_{1,i} \tag{3.69}$$

$$\pi_{di} = \frac{\sum_{t=1}^{T} y_{t,d}\mathbb{E}s_{t,i}}{\sum_{t=1}^{T} \mathbb{E}s_{t,i}}. \tag{3.70}$$

Note that the forward-backward algorithm is in calculated in the following manner. Firstly, the backward path computes the conditional probability of the observations $y_{t+1:T}$ given s_t as in (3.71) – (3.73):

$$\begin{aligned}
\beta_t &= P(y_{t+1:T}|s_t) && (3.71)\\
&= \sum_{s_{t+1}} P(y_{t+1:T}|s_{t+1})P(s_{t+1}|s_t)P(y_{t+1}|s_{t+1}) && (3.72)\\
&= \sum_{s_{t+1}} \beta_{t+1}P(s_{t+1}|s_t)P(y_{t+1}|s_{t+1}) && (3.73)\\
\beta_{t,i} &= P(y_{t+1:T}|s_t = i) && (3.74)
\end{aligned}$$

Secondly, the forward path computes α_t. This α_t is defined as the joint probability of s_t and the sequence of observations $y_{1:t}$.

$$\begin{aligned}
\alpha_t &= P(s_t, y_{1:t}) && (3.75)\\
&= \left[\sum_{s_{t-1}} P(s_{t-1}, y_{1:t-1})P(s_t|s_{t-1})\right] P(y_t|s_t) && (3.76)\\
&= \left[\sum_{s_{t-1}} \alpha_{t-1}P(s_t|s_{t-1})\right] P(y_t|s_t) && (3.77)\\
\alpha_{t,i} &= P(s_t = i, y_{1:t}) && (3.78)
\end{aligned}$$

Then, from these, we can compute the expections of $s_{t,i}$ and $s_{t,i}s_{t-1,j}$.

$$\begin{aligned}
\mathbb{E}s_{t,i} &= \gamma_{t,i} && (3.79)\\
&= \frac{\alpha_{t,i}\beta_{t,i}}{\sum_j \alpha_{t,j}\beta_{t,j}} && (3.80)\\
\mathbb{E}s_{t,i}s_{t-1,j} &= \xi_{t,i,j} && (3.81)\\
&= \frac{\alpha_{t-1,j}A_{i,j}P(y_t|s_{t,i})\beta_{t,i}}{\sum_{k,l} \alpha_{t-1,k}A_{k,l}P(y_t|s_{t,l})\beta_{t,l}} && (3.82)
\end{aligned}$$

From (3.55), we take logarithm in both sides as in (3.83):

$$\log P(s_{1:T}, y_{1:T}) \quad = \quad \log P(s_1) + \sum_{t=1}^{\top} \log P(y_t|s_t) + \sum_{t=2}^{\top} \log P(s_t|s_{t-1}) \tag{3.83}$$

We replace this representation with parameters $\theta = \{A, C, \pi\}$. Let $A_{(i,j)}$ be an element of the transition matrix of $K \times K$, which means the probability of transitioning from state j to state i. By this definition,

$$P(s_t|s_{t-1}) \quad = \quad \prod_{i=1}^{K} \prod_{j=1}^{K} (A_{ij})^{s_{t,i} s_{t-1,j}}. \tag{3.84}$$

We take the logarithm of both sides.

$$\log P(s_t|s_{t-1}) \quad = \quad \sum_{i=1}^{K} \sum_{j=1}^{K} s_{t,i} s_{t-1,j} \log A_{ij} \tag{3.85}$$

$$= \quad s_t^{\top} (\log A) s_{t-1}. \tag{3.86}$$

Let C be an emission matrix of size $D \times K$.

$$P(y_t|s_t) \quad = \quad C \tag{3.87}$$

$$\log P(y_t|s_t) \quad = \quad y_t^{\top} (\log C) s_t \tag{3.88}$$

$$\log P(s_1) \quad = \quad s_1^{\top} (\log \pi) \tag{3.89}$$

Altogether, we can rewrite (3.65) as in (3.90):

$$p(s_{1:T}, y_{1:T}|\theta) \quad = \quad s_1^{\top} \log C + \sum_{t=1}^{\top} s_t^{\top} \log C y_t \sum_{t=2}^{\top} \log s_{t-1}^{\top} \log A s_t \tag{3.90}$$

Note that we assume a homogeneous transition in the HMM word alignment model as in Definition 5.

Definition 5 (Homogeneous transition in HMM)**.** *When all the transitions in HMM from state s_u to s_v are position-independent, i.e. $p(A_i = s_u|A_{i=1} = s_v) = p(A_j = s_u|A_{j-1} = s_v)$, the HMM is called homogeneous. Notice that the alignment function $A_i = j$ means that f_j is mapped into $e_{a(i)}$, i.e. the j-th position of f is mapped into the i-th position of e. We consider only output values, i.e. $s_u = a(i)$ and $s_v = a(i-1)$.*

3.6.2 Standard HMM (standard ML): Graphical Model Implementation

The following description is based on (Bishop, 2006). Let f_n denote the factors and h denote the root node among such factors. These factors can be written as in (3.91):

$$\begin{cases} h(z_1) & = & p(z_1)p(x_1|z_1) \\ f_n(z_{n-1}, z_n) & = & p(z_n|z_{n-1})p(x_n|z_n) \end{cases} \tag{3.91}$$

First, we consider the messages from the leaf node to the root node. Using the sum-product algorithm, the messages can be writte as in (3.92):

$$\begin{cases} \mu_{z_{n-1}\to f_n}(z_{n-1}) & = & \mu_{f_{n-1}\to z_{n-1}}(z_{n-1}) \\ \mu_{f_n\to z_n}(z_n) & = & \sum_{z_{n-1}} f_n(z_{n-1}, z_n)\mu_{z_{n-1}\to f_n}(z_{n-1}) \end{cases} \tag{3.92}$$

Note that (3.92) is equivalent to the alpha recursions. From (3.92), we delete $\mu_{z_{n-1}\to f_n}(z_{n-1})$ to obtain a recursion from the factor f_n to the node z_n as in (3.93):

$$\mu_{f_n\to z_n}(z_n) = \sum_{z_{n-1}} f_n(z_{n-1}, z_n)\mu_{f_{n-1}\to z_{n-1}}(z_{n-1}) \tag{3.93}$$

If we define α as in (3.94):

$$\alpha(z_n) = \mu_{f_n\to z_n}(z_n), \tag{3.94}$$

this leads to the forward algorithm as in (3.95):

$$\alpha(z_n) \quad = \quad p(x_n|z_n) \sum_{z_{n-1}} \alpha(z_{n-1}) p(z_n|z_{n-1}) \tag{3.95}$$

Then, we consider the messages from the root node to the leaf node. Since variable nodes do not perform any computation, message can be writte as in (3.96):

$$\mu_{f_{n+1} \to f_n}(z_n) \quad = \quad \sum_{z_{n+1}} f_n(z_n, z_{n+1}) \mu_{f_{n+2} \to z_{n+1}}(z_{n+1}) \tag{3.96}$$

If we define β as in (3.97):

$$\beta(z_n) \quad = \quad \mu_{f_{n+1} \to z_n}(z_n), \tag{3.97}$$

this leads to the backward algorithm as in (3.98):

$$\beta(z_n) \quad = \quad \sum_{z_{n+1}} \beta(z_{n+1}) p(x_{n+1}|z_{n+1}) p(z_{n+1}|z_n) \tag{3.98}$$

Init message is shown in (3.99):

$$\mu_{z_N \to f_N}(z_N) \quad = \quad 1 \tag{3.99}$$

Note that the local marginal at the node z_n can be obtained by the product of the incoming messages as in (3.100):

$$p(z_n, X) \quad = \quad \mu_{f_n \to z_n}(z_n) \mu_{f_{n+1} \to z_n}(z_n) \tag{3.100}$$

$$= \quad \alpha(z_n) \beta(z_n) \tag{3.101}$$

Then, if we define $\gamma = p(z_n, X)/p(X)$, we obtain (3.103):

$$\gamma(z_n) = \frac{p(z_n, X)}{p(X)} \tag{3.102}$$

$$= \frac{\alpha(z_n)\beta(z_n)}{p(X)} \tag{3.103}$$

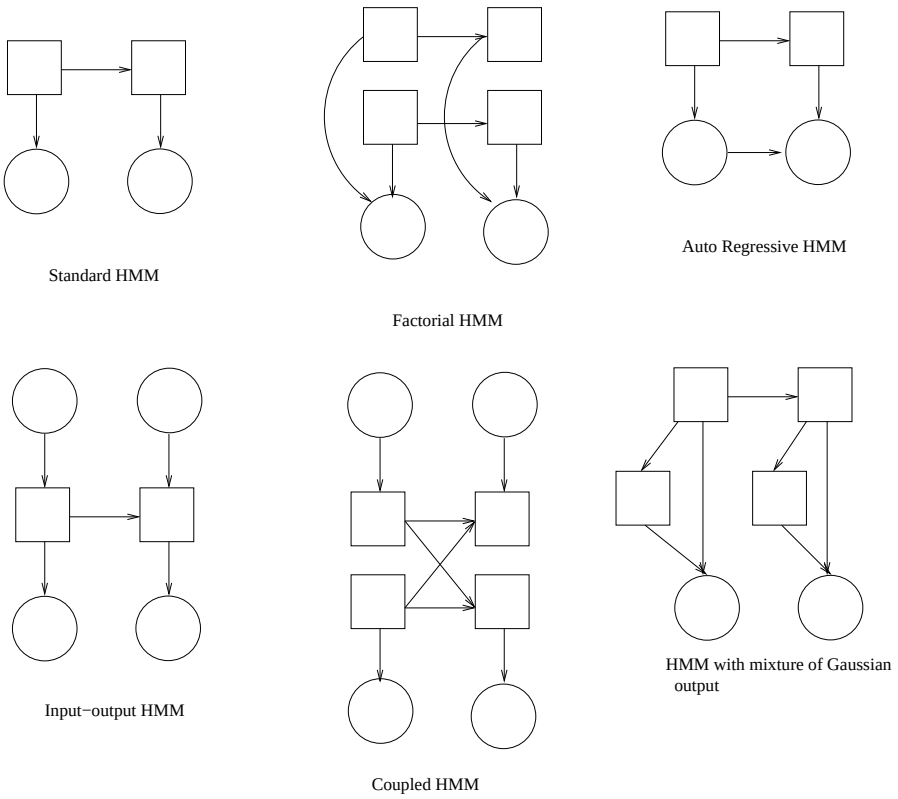

Figure 3.8: Five variants of HMM.

Figure 3.8 shows the standard HMM and five variants of HMM, whose network structure resembles the standard model. Traditionally, the algorithm of these need designing different algorithms. The graphical model facilitates the design of these each of the variants providing basic notions of inference / learning which can be reused. Hence, once we design an HMM, we can reuse it fairly easily when we implement a factorial HMM, for example.

3.6.3 Bayesian HMM (standard ML): Pseudo-count Implementation

A Bayesian HMM (Stolcke, 1994; Ghahramani, 2001; Beal, 2003) is one example of incorporating the prior into the standard HMM. It is often the case that the Dirichlet prior is selected due to its conjugacy. Dirichlet priors are applied on the transition and emission matrix, where we assume a multinomial distribution. The Dirichlet distribution is a conjugate prior over the Multinomial distribution.

Note that our main theme in this thesis is to embed the prior knowledge into an HMM which is slightly different from ours: the prior here is to prevent overfitting. Since excessive parameters in HMM cause overfitting, this method aims at regularising them using a prior to reduce such parameters, which is a common technique in HMM; then it maximises the a posteriori probability of the parameters by the MAP estimate.

The generative model can be written as in (3.105) $\sim$ (3.108):

$$\theta_t|\alpha \sim Dir(\alpha) \tag{3.104}$$

$$\phi_t|\alpha' \sim Dir(\alpha') \tag{3.105}$$

$$t_i|t_{i-1}=t \sim Multi(\theta_t) \tag{3.106}$$

$$w_i|t_i=t \sim Multi(\phi_t) \tag{3.107}$$

$$y \sim F(\phi) \tag{3.108}$$

For a sequence of draws $x = (x_1, \ldots, x_n)$ from a multinomial distribution θ with observed counts $n_1, \ldots, n_K$, a symmetric $Dirichlet(\beta)$ prior over θ yields the MAP estimate $\theta_k = \frac{n_k+\beta-1}{n+K(\beta-1)}$.

On the one hand, a multinomial distribution for unordered samples can be expressed as in (3.109):

$$P(c_1, \ldots, c_n|c, \theta) = \prod_{i=1}^{n} \theta_i^{c_i} \tag{3.109}$$

On the other hand, a Dirichlet prior distribution $g(a_q)$ can be expressed as in (3.110):

$$g(a_q) = \frac{1}{B(\nu_{1,q}, \ldots, \nu_{n,q})} \prod_{p=1}^{n} (q_{q,p})^{\nu_{q,p}-1} \tag{3.110}$$

where $B(\nu_{p,q}, \ldots, \nu_{n,q})$ is the Beta function in (3.111):

$$B(\nu_{1,q}, \ldots, \nu_{n,q}) = \frac{\Gamma(\nu_{1,q}) \ldots \Gamma(\nu_{n,q})}{\Gamma(\nu_{1,q} + \ldots + \nu_{n,q})} \tag{3.111}$$

where $\Gamma(n) = (n-1)!$. A Dirichlet prior is a conjugate prior, i.e. the same functional form as the likelihood function, of the multinomial distribution. Hence,

$$P(\theta|c_1, \ldots, c_n) = \frac{1}{B(\nu_{1,q} + \alpha_1, \ldots, \nu_{n,q} + \alpha_n)} \prod_{i=1}^{n} \theta_i^{\nu_{q,p}+\alpha_i-1}. \tag{3.112}$$

Then, integration of this yields a closed-form solution as in (3.114):

$$\int_\theta P(\theta) P(c_1, \ldots, c_n|\theta) d\theta = \frac{1}{B(\alpha_1, \ldots, \alpha_n)} \int_\theta \prod_{i=1}^{n} \theta_i^{c_i+\alpha_i-1} d\theta \tag{3.113}$$

$$= \frac{B(c_1 + \alpha_1, \ldots, c_n + \alpha_n)}{B(\alpha_1, \ldots, \alpha_n)} \tag{3.114}$$

Then the MAP estimate for $a_{q,p}$ can be expressed as in (3.115):

$$a_{q,p} = \frac{(\nu_{q,p} - 1) + c_{q,p}}{\sum_r (\nu_r - 1) + \sum_r c_{r,q}} \tag{3.115}$$

Note that this is equivalent to the addition of $(\nu_i - 1)$ virtual samples to the likelihood, which is called a pseudo count method.

3.6.4 Variational Bayesian HMM (standard ML)

This section follows the description of Beal (Beal, 2003). By considering the derivatives of the lower bound with respect to the variational posterior, it yields

$$\log q(s_{1:T}) \quad = \quad \langle \log p(s_{1:T}, y_{1:T}|\pi, A, C)\rangle_{q(\pi)q(A)q(C)} - \log \bar{z}(y_{1:T}). \tag{3.116}$$

where $z(y_{1:T})$ is a normalization constant. Using this, the complete-data likelihood is as in (3.118):

$$p(s_{1:T}, y_{1:T}) \quad = \quad p(s_1)p(y_1|s_1)\prod_{t=2}^{T} p(s_t|s_{t-1})p(y_t|s_t) \tag{3.117}$$

$$p(s_{1:T}, y_{1:T}|\theta) \quad = \quad p(s_1|\pi)\prod_{t=1}^{T} p(y_t|s_t, C)\prod_{t=2}^{T} p(s_t|s_{t-1}, A) \tag{3.118}$$

$$\log p(s_{1:T}, y_{1:T}|\theta) \quad = \quad \log p(s_1|\pi) + \sum_{t=1}^{T} \log p(y_t|s_t, C)\sum_{t=2}^{T} \log p(s_t|s_{t-1}, A) \tag{3.119}$$

$$= \quad \langle s_1^\top \log \pi + \sum_{t=1}^{T} s_t^\top \log C y_t + \sum_{t=2}^{T} \log s_{t-1}^\top \log A s_t\rangle_{q(\pi)q(A)q(C)}$$
$$- \log \bar{z}(y_{1:T}) \tag{3.120}$$

$$= \quad s_1^\top \langle \log \pi\rangle_{q(\pi)} + \sum_{t=1}^{T} s_t^\top \langle \log C\rangle_{q(C)} y_t + \sum_{t=2}^{T} s_{t-1}^\top \langle \log A\rangle_{q(A)} s_t$$
$$- \log \bar{z}(y_{1:T}) \tag{3.121}$$

Now we consider the parameter θ as well as the natural parameter vector $\phi(\theta)$ given in (3.123):

$$\theta \quad = \quad (\pi, A, C) \tag{3.122}$$

$$\phi(\theta) \quad = \quad (\log \pi, \log A, \log C) \tag{3.123}$$

Note that the expectation of the natural parameters ϕ and θ are given as in (3.125):

$$\bar{\phi} = \langle \phi(\theta) \rangle_{q(\theta)} \tag{3.124}$$

$$= (\langle \log \pi \rangle_{q(\pi)}, \langle \log A \rangle_{q(A)}, \langle \log C \rangle_{q(C)}) \tag{3.125}$$

and

$$\bar{\theta} = \phi^{-1}(\langle \phi(\theta) \rangle_{q(\theta)}) \tag{3.126}$$

$$= (\exp\langle \log \pi \rangle_{q(\pi)}, \exp\langle \log A \rangle_{q(A)}, \exp\langle \log C \rangle_{q(C)}) \tag{3.127}$$

$$= (\bar{\pi}, \bar{A}, \bar{C}). \tag{3.128}$$

Using the standard results shown in (3.129):

$$\int d\pi Dir(\pi|u) \log \pi_j = \psi(u_j) - \psi(\sum_{j=1}^{k} u_j) \tag{3.129}$$

we can compute the expectations of the logarithm of the parameters under the Dirichlet distributions as in (3.132):

$$\bar{\pi} = \exp\left[\psi(w_j^{(\pi)} - \psi(\sum_{j=1}^{k} w_j^{(\pi)})\right] : \sum_{j=1}^{k} \bar{\pi_j} \leq 1 \tag{3.130}$$

$$\bar{A} = \exp\left[\psi(w_{jj'}^{(A)} - \psi(\sum_{j=1}^{k} w_{jj'}^{(A)})\right] : \sum_{j'=1}^{k} \bar{a_{jj'}} \leq 1 \forall j \tag{3.131}$$

$$\bar{C} = \exp\left[\psi(w_{jm}^{(C)} - \psi(\sum_{j=1}^{p} w_{jm}^{(C)})\right] : \sum_{m=1}^{p} \bar{c}_{jm}^{k} \bar{a_{jj'}} \leq 1 \forall j \tag{3.132}$$

As is similar with the standard HMM, we use the Baum-Welch algorithm (or forward-backward

algorithm) to derive α and β, as in (3.134):

$$\alpha_t(s_t) = \frac{1}{\bar{\xi}_t(y_t)} \left[\sum_{s_t=1} \alpha_{t-1}(s_{t-1}) \bar{p}(s_t|s_{t-1}) \right] \bar{p}(y_t|s_t) \tag{3.133}$$

$$\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) \bar{p}(s_{t+1}|s_t) \bar{p}(y_{t+1}|s_{t+1})) \tag{3.134}$$

The calculation of the lower bound is done in the following manner. The product of normalisation constants is as in (3.135):

$$\prod_{t=1}^{T} \bar{\xi}_t(y_t) = \bar{z}(y_{1:T}). \tag{3.135}$$

Then, we can write as in (3.136):

$$F(q(\pi, A, C), q(s_{1:T})) = \int d\pi q(\pi) \log \frac{p(\pi)}{q(\pi)} + \int dA q(A) \log \frac{p(A)}{q(A)} + \int dC q(C) \log \frac{p(C)}{q(C)} + H(q(s_{1:T})) + \langle \log p(s_{1:T}, y_{1:T}|\pi, A, C) \rangle_{q(\pi)q(A)q(C)q(s_{1:T})} \tag{3.136}$$

where $H(q(s_{1:T}))$ denotes the entropy of the variational posterior distribution over hidden state sequences, as in (3.137):

$$H(q(s_{1:T})) = -\sum_{s_{1:T}} q(s_{1:T}) \langle \log p(s_{1:T}, y_{1:T}|\pi, A, C) \rangle_{q(\pi)q(A)q(C)} + \log \bar{z}(y_{1:T}) \tag{3.137}$$

Using this, the computation of the lower bound $F(q(\pi, A, C), q(s_{1:T}))$ involves the evaluation of KL divergences between variational posterior and prior Dirichlet distributions for each row of π, A, C and the normalisation constants $\{\xi_t(\bar{y}_t)\}_{t=1}^{T}$, which is in (3.138):

$$F(q(\pi, A, C), q(s_{1:T})) = \int d\pi q(\pi) \log \frac{p(\pi)}{q(\pi)} + \int dA q(A) \log \frac{p(A)}{q(A)} + \int dC q(C) \log \frac{p(C)}{q(C)} + \log \bar{z}(y_{1:T}) \tag{3.138}$$

M-step:

We take the functional derivatives of F with respect to each of these distributions and consider the point which takes zero. This yields the following Dirichlet distributions.

$$
\begin{aligned}
q(\pi) &= Dir(\{\pi_1, \ldots, \pi_k\} | \{w_1^{(\pi)}, \ldots, w_k^{(\pi)}\}), w_j^{(\pi)} \\
&= u_j^{\pi} + \langle \delta(s_1, j) \rangle_{q(s_{1:T})} \qquad (3.139) \\
q(A) &= \prod_{j=1}^{k} Dir(\{a_{j1}, \ldots, a_{jk}\} | \{w_{j1}^{(A)}, \ldots, w_{jk}^{(A)}\}), w_{jj'}^{(A)} \\
&= u_{j'}^{A} + \sum_{t=2}^{T} \langle \delta(s_{t-1}, j) \delta(s_t, j') \rangle_{q(s_{1:T})} \qquad (3.140) \\
q(C) &= \prod_{j=1}^{k} Dir(\{c_{j1}, \ldots, c_{jk}\} | \{w_{j1}^{(C)}, \ldots, w_{jk}^{(C)}\}), w_{jq}^{(C)} \\
&= u_q^{C} + \sum_{t=1}^{T} \langle \delta(s_t, j) \delta(s_t, q) \rangle_{q(s_{1:T})} \qquad (3.141)
\end{aligned}
$$

3.7 Algorithmic Design: Inference

While Sections 3.5 and 3.6 derives lexical translation probabilities (Section 3.5) or HMM Model (Section 3.6), this section derives a Viterbi alignment using the obtained lexical translation probabilities or models. Viterbi alignment is the most likely sequence of alignments for a particular sentence pair (Refer to the righthand side of upper and lower figures in Figure 1.3). Although the importance of this process is seldom described, this process is inevitable in order to pipe the obtained lexical translation probabilities into the form of Viterbi alignment. Viterbi alignment of the training corpus is the input to the phrase extraction process.

3.7.1 Viterbi Decoding (standard ML, GIZA++)

Let us define an observed sequence X, a model parameter θ, and latent variables t. If we are interested in obtaining the posterior distribution for a given observation, we are to seek $P(\theta|X)$. On the other hand, the quantity $P(t = 1|\theta, X)$ is sometimes useful when the model is sensitive to

the choice of t, which is often the case. The former corresponds to the Viterbi decoding (Viterbi, 1967) and the latter does to the posterior decoding.

A Viterbi decoding is needed to obtain the most probable state path for a given model $\mathcal{M}$.

Definition 6 (Viterbi decoding)**.** *Given HMM model $\mathcal{M}$ and a observed sequence $X_1, \ldots, X_n$, find the most likely sequence of states $s_1, \ldots, s_n$ which could have generated this observed sequence.*

3.7.2 Viterbi Decoding (standard ML): Graphical Model Implementation

The following description is based on (Bishop, 2006). A factor graph for HMM is shown in Figure 3.9. Let the node z_n denote the root. We pass massages from the leaf nodes to the root using the max-sum algorithm, where the messages in the max-sum algorithm are shown as in (3.142):

$$\begin{cases} \mu_{z_n \to f_{n+1}} & = & \mu_{f_n \to z_n}(z_n) \\ \mu_{f_{n+1} \to z_{n+1}} & = & \max_{z_n}\{\ln f_{n+1}(z_n, z_{n+1}) + \mu_{z_n \to f_{n+1}}(z_n)\} \end{cases} \tag{3.142}$$

By eliminating $\mu_{z_n \to f_{n+1}}(z_n)$ from (3.142), we obtain a recursion form the message from f to z as in (3.143):

$$\omega(z_{n+1}) \quad = \quad \ln p(x_{n+1}|z_{n+1}) + \max_{z_n}\{\ln p(x_{+1}|z_n) + \omega(z_n)\} \tag{3.143}$$

where $\omega(z_n) = \mu_{f_n \to z_n}(z_n)$. Note that init message is shown as in (3.144):

$$\omega(z_1) \quad = \quad \ln p(z_1) + \ln p(x_1|z_1)) \tag{3.144}$$

Note that the Viterbi algorithm can also be obtained from the joint distribution. We take the logarithm and exchange $\max$ and $\sum$, which is shown as in (3.145):

$$\omega(z_n) \quad = \quad \max_{z_1,\ldots,z_{n-1}} p(x_1, \ldots, x_n, z_1, \ldots, z_n) \tag{3.145}$$

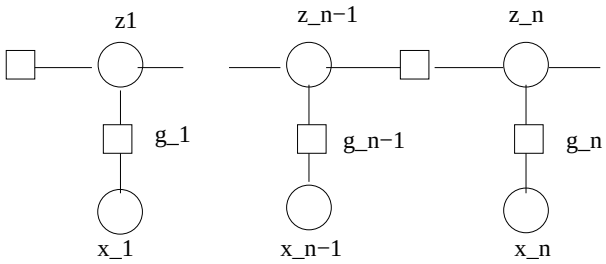

Figure 3.9: Figures shows a factor graph of hidden Markov model.

As is explained in Section 2.3. on the max-product (max-sum) algorithm, once we computed the $\max$ operations, we can obtain the most probable path. We can also obtain the state sequences corresponding to this path by the back-tracking procedure.

$$k_n^{max} \quad = \quad \phi(k_{n+1}^{max}) \tag{3.146}$$

If we only keep track of K possible paths in K current states, we have K^2 possible paths. However, at time $n+1$, we only retain K of these paths corresponding to the best path for each state. At step N, we discover the most probable path. Then, since there is a unique path which uses the state in its path, it is possible to back-track the path.

3.7.3 Posterior Decoding (standard ML)

A posterior decoding is to obtain multiple paths which have a high probability with thresholding the posterior probability δ in each state. This contrasts with Viterbi decoding where the single most probable path is obtained. Let us define a posterior state probability $P(X_1, \ldots, X_n, s_t = i)$, namely a probability that an HMM model $\mathcal{M}$ generates the sequence $X_1, \ldots, X_n$ passing through the state i at time t. Then, the aim of a posterior decoding is for each $i(1 \leq i \leq n)$ to find a state t_i that is the most likely to have generated the symbol at that position, as in Algorithm 1:

Definition 7 (Posterior decoding)**.** *Given HMM model $\mathcal{M}$ and sequence $X = X_1, \ldots, X_n$, for each position $1 \leq i \leq n$, find the state t_i that most likely to have generated the symbol at that*

position.

Algorithm 1 Pseudocode for posterior decoding

```
def posteriorDecoding(e,f,t,forward,backward,threshold)
Step 1: (e,f)=getSentence()
Step 2: forward=makeForward(e,f,t)
Step 3: backward=makeBackward(e,f,t)
Step 4: likelihood = makeLikelihood(forward,backward)
Step 5: posterior = makePosterior(forward,backward,likelihood)
Step 6: for e_i in e:
Step 7:         for f_j in f:
Step 8:             alignment.addPosteior(e_i,f_i, posterior[e_i,f_j])
Step 9:             if (posterior[e_i,f_j] > threshold):
Step 10:                alignment.add([e_i,f_j])
```

3.8 Data Design: Training Data Manipulation

In Machine Learning research, there are various data manipulation techniques, such as data centering, data standardization, anomality detection (or noise reduction), principle component analysis (PCA), singular value decomposition (SVD), ANOVA analysis, lower-dimensional reduction, and so forth. This section focuses on noise reduction (or anomality detection).

We describe algorithms to handle both sentence-level and word-level noise. At the sentence-level, we first describe an hueristic sentence cleaning algorithm, then move to the bigram-based sentence cleaning algorithm. At the word-level, we describe two kinds of noise-sensitive MAP-word aligner: this word aligner can inherently handle noise if it is detected beforehand. The first MAP-based word aligner is simple where prior knowledge is consistent, while the second MAP-based word aligner is complex where prior knowledge is not consistent. With this MAP-based word aligner, we aim at handling three kinds of noise: 1) many-to-many mapping objects, 2) translational noise, and 3) noisy mapping between function words and content words. The first one is relatively easy to handle, except that the computational complexity is high. The third one, the detection of translational noise, is very difficult as unfortunately, we cannot use the results of word alignment

to judge translational noise. Figure 3.10 shows that the 0-to-1 mappings amount to around 30% in the EN to DE direction. In reality, translational noise is fairly low in number between European languages, say less than 1%. A null alignment does not suggest that this immediately indicates translational noise.

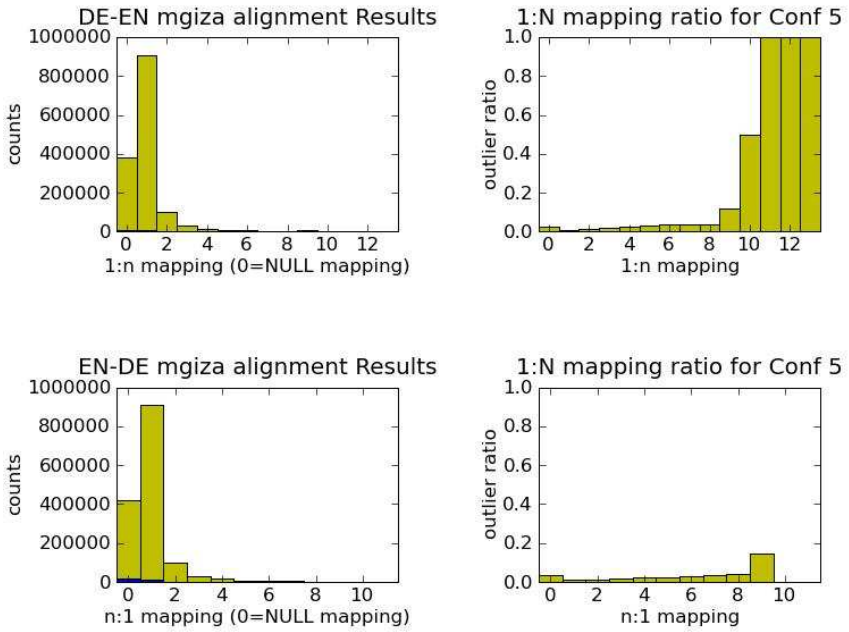

Figure 3.10: Figures A and C show the results of word alignment for DE-EN where outliers detected by Algorithm 1 are shown in blue at the bottom. We check all the alignment cept pairs in the training corpus inspecting so-called A3 final files by type of alignment from 1:1 to 1:13 (or NULL alignment). It is noted that outliers are miniscule in Figures A and C because each count is only 3%. Most of them are NULL alignments or 1:1 alignments, while there are small numbers of alignments with 1:3 and 1:4 (up to 1:13 in the DE-EN direction in Figure A). In Figure C, 1:11 is the greatest. Figures B and D show the ratio of outliers over all the counts. Figure B shows that in the case of 1:10 alignments, 1/2 of the alignments are considered to be outliers by Algorithm 1, while 100% of alignments from 1:11 to 1:13 are considered to be outliers (false negatives). Figure D shows that in the case of EN-DE, most of the outlier ratios are less than 20%.

Note that the effectiveness of these algorithms depends on the particular situation, especially in its level of noise and its language-specific noise arising from the corpus. Apart from the level of noise which depends on the corpus, the difference of language between JP-EN is bigger than

X	ENFR	FREN	ESEN	DEEN	ENDE
10	0.098	0.096	0.143	0.097	0.079
20	0.165	0.165	0.246	0.138	0.127
30	0.193	0.187	0.279	0.157	0.137
40	0.201	0.199	0.295	0.168	0.142
50	0.208	0.201	0.297	0.170	0.145
60	0.211	0.203	0.297	0.171	0.146
70	0.212	0.202	0.298	0.170	0.146
80	0.211	0.202	0.301	0.169	0.147
90	0.212	0.202	0.297	0.171	0.147
100	0.211	0.202	0.302	0.169	0.146
#	43k	43k	51k	60k	60k
ave len	21.0/23.8(EN/FR) 20.9/24.5(EN/ES) 20.6/21.6(EN/DE)				

Table 3.4: BLEU score after cleaning of sentences with length greater than X. The row shows X, while the column shows the language pair. Parallel corpus is News Commentary parallel corpus (WMT07).

between EN-FR, therefore let us focus on the NTCIR JP-EN data rather than Europarl between EN-FR.

3.8.1 Sentence-level Corpus Cleaning (SMT oriented, Moses)

A heuristic sentence-level cleaning algorithm is shown in Algorithm 2 whose code is provided in Moses.

Algorithm 2 Sentence Cleaning Algorithm (Heuristic in state-of-the-art SMT)

Remove sentences with lengths greater than X (or remove sentences with lengths smaller than X in the case of short sentences).

This algorithm reduces training sentence pairs based on the indirect feature *sentence length*. Interestingly as is shown in Table 3.4 this algorithm may often improve the BLEU score.

Figure 3.11 gives us some insights as to why this algorithm improves BLEU score, since this algorithm removed the region where the outlier ratio is high in this example. The three figures on the right show that if we view this using the ratio of outliers over all of the sentence-cleaning, all

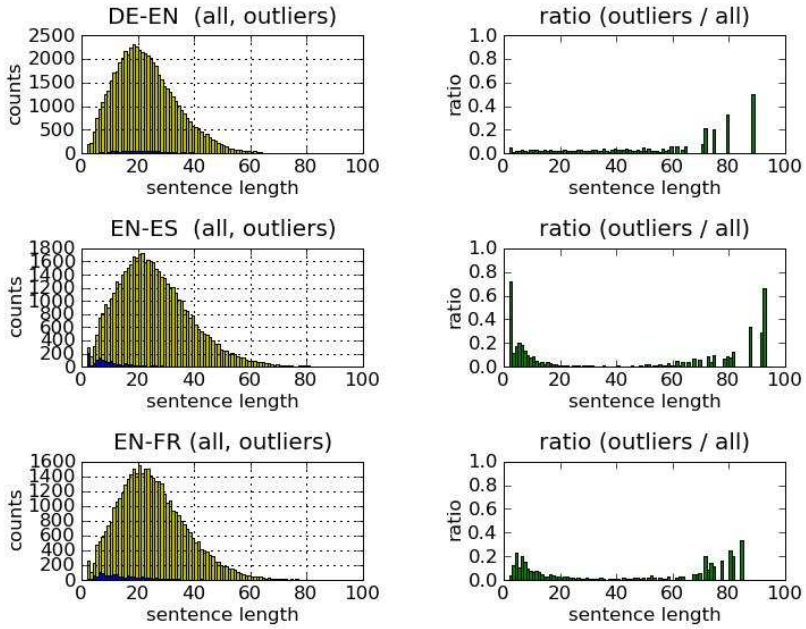

Figure 3.11: The three figures on the left show histograms of the number of sentences over sentence length. The three figures on the right show histograms of the outlier ratio over sentence length. This histogram shows that the region whose sentence length is more than 70 or 80 (in the lower two figures, the region whose sentence length is between 1 and 10 as well) has higher ratio of outliers compared to other area. This explains why Algorithm 2 works. Note that outliers in this figure were detected by Algorithm 3.

three figures tend to have more than 20–30% of their sentences having a length of 80–100 words. The lower two figures show that sentence length 1 to 4 tends to be less than 10% of the figure. (As the numbers of outliers are less than 5% in each case, the outliers are miniscule. In the case of EN-ES, we can observe the dark blue small distributions at the bottom of each figure from sentence length 2 to 16.)

3.8.2 Sentence-level Outlier Detection (original)

This algorithm comes from the analogy of the outlier detection. In a linear regression problem, it is known that the existence of a few extreme points called outliers which are distant from the decision plane can drastically affect results. If this is the case, we would obtain better results by

removing such outliers, or conversely by collecting good inliers. We show this algorithm in the context of SMT to make a good point algorithm.

First we define the noise detection problem as in Definition 8.

Definition 8 (Noise detection)**.** *Let $S = \{(\breve{e}_1, \breve{f}_1), \ldots, (\breve{e}_n, \breve{f}_n)\}$ be a parallel corpus consisting of a training and development corpus. For a given parallel corpus, the noise detection task is to detect sentence pairs $(\breve{e}_i, \breve{f}_i)$ that contain noise (or unfavourable elements). Note that unfavourable elements include many-to-many mapping objects.*

One difficulty is that we have no appropriate direct measure to assess the quality of word alignment due to the unavailability of hand-annotated data with alignment links; AER (Och and Ney, 2003) can be applied only if a hand-annotated corpus exists and if in-domain test data is available. With BLEU as an evaluation measure, the approach taken in (Okita, 2009a) is as follows. Let $S = \{(\breve{e}_1, \breve{f}_1), \ldots, (\breve{e}_n, \breve{f}_n)\}$ be a training corpus and let $M : \breve{f} \rightarrow \breve{e}$ be our MT system trained on this training corpus S. If the distance between a reference translation $\breve{e}_{\bar{i}}$ and $M(\breve{f}_i)$ is big for relatively small data sets, this may indicate that the sentence $\breve{f}_i$ is relatively difficult to translate; this may be due to a training sentence $\breve{f}_i$ being too complex for the model complexity of MT system M. By removing the detected noisy sentences we reduce the training corpus and rerun the word aligner.[8] In sum, this algorithm becomes as in Algorithm 3.

[8]It is to be noted that we set aside the problem of whether this approach actually improves the test set accuracy. It is also noted that if our parallel corpus is sufficiently big, we would use the held-out datasets to validate the results. If our parallel corpus is small, this would not be possible since Step 5 seeks the points whose score is zero.

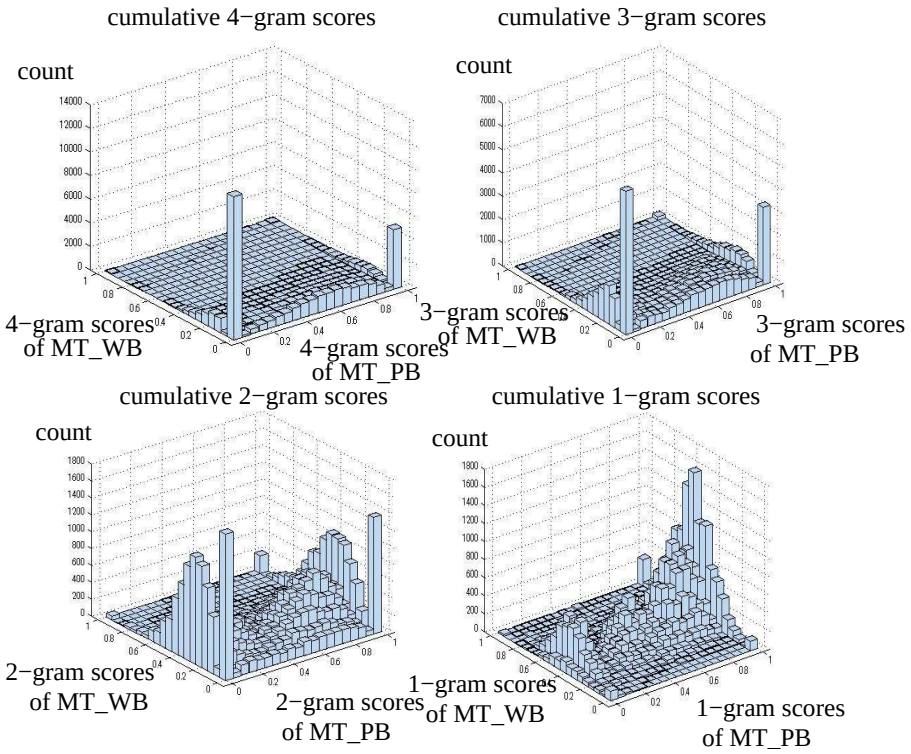

Figure 3.12: The sentence-based cumulative n-gram scores: x-axis is phrase-based SMT and y-axis is word-based SMT. The focus is on the worst point (0,0) where both scores are zero. Many points reside in (0,0) in cumulative 4-gram scores, while only small numbers of points reside in (0,0) in cumulative 1-gram scores.

Algorithm 3 Revised Good Points Algorithm

Step 1: Train word-based MT (Moses with MAX__PHRASE__LENGTH = 1) on the full parallel corpus. Translate all training sentences by the above mentioned word-based MT decoder.

Step 2: Obtain the cumulative X-gram score $S_{WB,X}$ for each pair of sentences where X is 4, 3, 2, and 1 for word-based MT decoder.

Step 3: Train PB-SMT on the full parallel corpus. Note that we do not need to run a word aligner again here, but use the results of Step 1. Translate all training sentences by the above mentioned PB-SMT decoder.

Step 4: Obtain the cumulative X-gram score $S_{PB,X}$ for each pair of sentences where X is 4, 3, 2, and 1 for PB-SMT decoder.

Step 5: Remove sentences whose $(S_{WB,2},\, S_{PB,2}) = (0, 0)$. We produce new reduced parallel corpus.

(Step 6: Do the whole procedure of PB-SMT using the reduced parallel corpus which we obtain from Step 1 to 5.)

Note that the sentence duplication algorithm (Okita, 2009b) shown in Algorithm 4 is not to reduce the number of sentences in the training corpus, but to duplicate particular sentences. It seems that this works comparably well or even better in some cases, but this invokes some minor problems in GIZA++, which cause computation to not be completed.

Algorithm 4 Sentence Duplication Algorithm

Step 1: Conditioned on a sentence length pair (l_e,l_f), we count the numbers of them. We calculate the ratio $r_{i,j}$ of this number over the number of all sentences.
Step 2: If this ratio $r_{i,j}$ is under the threshold X, we duplicate N times.

One way to explain the rationale behind this algorithm is in the power-law distribution. Since this algorithm allows an increase in the number of sentences in the tail of this distribution, the distribution at the tail is amplified. Note that the power-law distribution is also called a long tail distribution where there are many elements whose counts are scarce.

3.8.3 Word-level Noise-Sensitive MAP-based Word Aligner (original)

Given that a parallel corpus contains unlabeled many-to-many mapping objects (the usual case in MT), our first target to solve is the following problem in Definition 9.

Definition 9 (many-to-many mapping object detection)**.** *Let* $S = \{(\breve{e}_1, \breve{f}_1), \ldots, (\breve{e}_n, \breve{f}_n)\}$ *be a parallel corpus. For a given parallel corpus, many-to-many mapping object detection is the task of detecting sentence pairs* $(\breve{e}_i, \breve{f}_i)$ *which include many-to-many mapping objects, such as paraphrases, non-literal translations, and multi-word expressions.*

If noisy objects were limited within many-to-many mapping objects. the experimental results based on Definition 8 and Definition 9 would be similar: we may call Definition 8 an intrinsic method and Definition 9 an extrinsic method in this case. The extrinsic method does not detect sentences that include many-to-many mapping objects in a straightforward way, but instead detects them via indirect measures that identify sentences that contain many-to-many mapping objects. Note that depending on the results of word alignment, the phrase extraction heuristics, such as

grow-diag-final, are capable of extracting many-to-many mapping objects. We also note that it turns out (Refer to the latter part of Section 3.10.3) that the assumption above is not correct, but we have various other noisy objects as well. Our current estimate of the list of noisy objects is shown in Section 3.1.3.

The intrinsic method identifies the many-to-many mapping objects themselves. Unfortunately, many-to-many mapping objects are still at large which include at least three cases such as paraphrases, non-literal translations, and multi-word expressions. We limit it here to the NP / MWE detection task which is defined as follows:

Definition 10 (Bilingual NP detection)**.** *Let* $S = \{(\breve{e}_1, \breve{f}_1), \ldots, (\breve{e}_n, \breve{f}_n)\}$ *be a parallel corpus consisting of a training and development corpus. For a given parallel corpus, the bilingual NP detection task is to detect NPs in each sentence* $(\breve{e}_i, \breve{f}_i)$*.*

The purpose of the bilingual NP detection task is to identify the *anchor words* [9] defined as in Definition 11. Note that we suppose here that we consider each NP as one word. Then we can handle them in the same way as if we see a pair of words in word alignment.

Definition 11 (Anchor Words)**.** *Let* $(\breve{e}, \breve{f}) = \{(\breve{e}_1, \breve{f}_1), \ldots, (\breve{e}_n, \breve{f}_n)\}$ *be a parallel corpus. By prior knowledge we additionally have knowledge of anchor words* $(\hat{e}, \hat{f}) = \{(sent_i, t_{e_1}, t_{f_1}, pos_{e_1}, pos_{f_1}, length_e, length_f), \ldots, (sent_k, t_{e_n}, t_{f_n}, pos_{e_n}, pos_{f_n}, length_e, length_f)\}$ *where* $sent_i$ *denotes sentence ID,* pos_{e_i} *denotes the position of* t_{e_i} *in a sentence* $\breve{e}_i$*, and* $length_e$ *(and* $length_f$*) denotes the sentence length of the original sentence which includes* e_i*. We call such prior knowledge for alignment links "anchor words".*

Hence, using anchor words [10] we invoke the MAP-based word aligner which we described in Section 3.4 of MAP-EM and 3.5 of Bayesian HMM. This task is defined as in Definition 12:

[9] This term is used by Church and Gale (Gale and Church, 1991) in the context of sentence alignment.
[10] One example of anchor words is a pair of NP. Other example is paraphrases shown in Table 1.2.

Definition 12 (Noise-sensitive MAP-based word alignment)**.** *Given anchor words defined above and a parallel corpus, we align words using the MAP-based word aligner (Refer also to Sections 3.4 MAP-EM and 3.5 Bayesian HMM).*

3.9 Linguistic Domain Knowledge about Word Alignment Links

This section describes possible techniques to extract some linguistic domain knowledge which is required to pursue the approach described in the subsection on the word-level noise-sensitive MAP-based word aligner: this approach is to detect linguistic domain knowledge in the first step, and then incorporate such linguistic domain knowledge as prior knowledge to the MAP-based word aligner.

In the sections on Algorithmic designs, we have discussed the machine learning algorithms, EM-based algorithms and HMM-based algorithms. The EM-based algorithms in Section 3.4 do not consider any dependencies, while HMM-based algorithms in Section 3.5 consider only the first-order Markov dependencies. This section intends to introduce more varieties of dependencies in bilingual settings.

Our MAP-based word aligner may use various knowledge sources in order to set up the prior. The classical examples of knowledge source includes orthographic features, POS tags, bilingual dictionary, Markov features, relative sentence position, the maximum translation score between the source and one of the target words (null), and dice and Model 1 (Moore et al., 2006; Blunsom and Cohn, 2006). We may also collect more direct linguistic knowledge for word alignment, which will result in higher precision: lexical semantics derived from a bilingual wordnet, multi-word expressions, functional word sequence patterns (which is similar to synchronous grammatical knowledge), reference lists derived by co-reference resolution together with zero anaphora resolution, named entities as established by a named-entity recognizer, markers which suggest information structure, dependency relations via dependency parser, and Out-Of-Vocabulary (OOV) words. [11]

[11] Note that this section scratches only the several topics. There would be many other linguistic knowledge which

3.9.1 NPs / MWEs / Idioms (Structured Relation: Lexicon)

Non-compositionality concerns the case in which the meaning of a constituent is not based on the meaning of its parts (Jurafsky and Martin, 2009). An example is the phrase *the tip of the iceberg* in the next sentence: Coupons are just *the tip of the iceberg* (non-compositional word). This suggests that we need to differentiate between the fragments of this phrase and the phrase itself in word alignment. The exact treatment of this matter will be related to detect compositionality in each side of languages independently. However, due to the difficulty of detecting such compositionality in practice, our method only concerns this issue as a part of the bilingual terminology extraction process: the terminology extracted by our algorithm contains not only non-compositional terminology, but both compositional and non-compositional terminology. We extract NPs by the method of Kupiec (Kupiec, 1993) and paraphrases by the method of Callison-Burch et al. (Callison-Burch, 2008). Unfortunately, lexical resources, such as Wordnet, do not help much as their NPs are small in number.

One algorithm that we used for extracting NPs is a statistical method which is a bidirectional version of Kupiec (1993). Firstly, Kupiec presents a method to extract bilingual noun phrase pairs in a unidirectional manner based on the knowledge about typical POS patterns of noun phrases, which is language-dependent but can be written down with some ease by a linguistic expert. For example in French they are N N, N prep N, and N Adj. Secondly, we take the intersection (or union) of extracted bilingual noun phrase pairs.[12]

will be the source of word alignment links. However, we left these as a further study.

[12] In word alignment, bidirectional word alignment by taking the intersection or union is a standard method which improves its quality compared to unidirectional word alignment.

Algorithm 5 Noun Phrase Extraction Algorithm

Given: a parallel corpus and a set of anchor word alignment links:
1. We use a POS tagger (Part-Of-Speech Tagger) to tag a sentence on the SL side.
2. Based on the typical POS patterns for the SL, extract noun phrases on the SL side.
3. Count n-gram statistics (typically $n = 1, \cdots, 5$ are used) on the TL side which jointly occur with each source noun phrase extracted in Step 2.
4. Obtain the maximum likelihood counts of joint phrases, i.e. noun phrases on the SL side and n-gram phrases on the TL side.
5. Repeat the same procedure from Step 1 to 4 reversing the SL and TL.
6. Intersect (or union) the results in both directions.

Let SL be the source language side and TL be the target language side. The procedure is shown in Algorithm 5. We informally evaluated the noun phrase extraction tool following Kupiec (1993) by manually inspecting the mapping of the 100 most frequent terms. For example, we found that 93 of the 100 most frequent English terms in the patent corpus were correctly mapped to their Japanese translation.

The second method is to use domain knowledge about implicit alignment links, which can be only applicable to specific parallel corpora such as patent and technical document corpora, we can use heuristics to extract the "noun phrase" + "reference number" from both sides. This is due to the fact that terminology is often labelled with a unique reference number, which is labelled on both the SL and TL sides.

3.9.2 Translational Noise

The term 'translational noise' is often used in a context of human translator who adds or removes words only on one side in order to clarify the semantics. Suppose that we translate a Japanese word 'tsunami' into English. Since there is no appropriate corresponding word in English (at least until recently), a human translator may add something else written in the text in order to explain the semantics of this word in English. For example, other than the translation of this word such as 'a big wave' or 'a series of water waves caused by the displacement of a large volume of a body of water' (by Wikipedia), a human translator may add 'For example, Sumatra was hit by a

tsunami in 2004', which is not written anywhere in the source. The human translator recognizes that the translation of the target side will not be equivalent information if he / she performs back translation. However, a human translator needs to convey the semantics of the source side, which often results in superfluous elements on the target side. However, we do not include such cases as 'translational noise' simply because this seems to be a very difficult problem by a word aligner.

Instead of handling this difficult problem, we intend here to handle only simpler cases. For example in Japanese, phrases such as "monodearu", "youninaru", "monotonaru", "youninatteiru", and "kototonaru" often exist in the written text, but mean nothing. (These phrases can be deleted from the Japanese sentences without loss of meaning.) Even these phrases are existed or deleted, the coresponding English sentence will not change. However, their existence does affect the word aligner since one to five words are created from such phrases depending on the morphological analyzer. It seems that it is not very easy to show such examples between European languages. However, consider the sentence pair ('on this **particular** building','*dans ce bâtiment*') from Europarl (Koehn, 2005). We call the word **particular** *noise* since it only exists on the English side and may cause problems for a word aligner.

Table 3.5 shows the motivation of our study that BLEU score is improved by 0.70 to 1.0 points (PB-SMT) by removing only five typical Japanese phrases "monodearu", "youninaru", "monotonaru", "youninatteiru", and "kototonaru". Note that we realize that the detection of such translational noise is very difficult. For example, it will be misleading if we look at the results of word alignment since 0-to-1 mapping objects amount to around 30%, which should not be the case.

	ENJP	JPEN
200k PB-SMT (baseline)	24.96	17.96
200k PB-SMT (removed)	25.69	18.96
200k HPB-SMT (baseline)	28.51	21.89
200k HPB-SMT (removed)	27.93	21.94

Table 3.5: Translational noise only removing five typical redundant phrases from the Japanese side of the training corpus.

0:when:
1:the:
2:fluid:[サブスタンス,リキッド,体液,動体,汁,流体,流動体,流動物,液,液体,物,物質]
3:pressure:[うしろ押し,エマージェンシー,プッシュ,プッシング,プレス,プレッシャー,一押し,体性感覚,切迫,力,単位面積当たりの力,危急,危殆,危難,圧し,圧力,圧覚,圧迫,後ろ押し,後押,後押し,急場,急迫,押,押し,押すこと,気圧,火急,物理現象,究追,空圧,窮状,窮迫,窮追,緊急,重圧]
4:cylinder:[コンテナ,コンテナー,シリンダ,シリンダー,ソリッド,大夫,入れもの,入れ物,入物,円柱,円筒,器,器物,固体,固形,容れもの,容れ物,容器,容物,益荒男,部屋]
5:@card@:
6:be:[(omitted)]
7:use:[使い果たす,使う,使用,働かす,働かせる,充てる,利かす,利する,利用,動く,取り入れる,取入れる,実行,実践,実践躬行,履行,引き当てる,引当てる,引当る,当てはめる,当てる,当て嵌める,役する,役だてる,役立てる,悪用,摂する,摂る,摂取,服する,服用,活用,消費,用いる,経口摂取,行ずる,行なう,行使,行動,運用,適用]
8:,:
9:fluid:[サブスタンス,リキッド,体液,動体,汁,流体,流動体,流動物,液,液体,物,物質]
10:be:[(omitted)]
11:gradually:[おもむろに,じょじょに,じわじわ,じわっと,じわり,じんわり,ちびちび,ちびりちびり,ひたひた,ぼちぼち,ぼつぼつ,ぼちぼち,ぼつぼつ,ゆっくり,ジワジワ,垂々,垂垂,少しずつ,徐々,徐々に,徐に,徐徐,徐徐に,次第,次第に,次第次第,次第次第に,歩々,歩歩,段段,漸々,漸う,漸く,漸次,漸漸,漸進的,追々,追い追い,追い追いに,追追]
12:apply:[あてはまる,くれてやる,つける,リクエスト,付ける,使う,使用,係る,係わる,働かす,働かせる,充てる,分け与える,分配,列なる,利かす,利する,利用,塗る,実行,実践,実践躬行,尽くす,履行,差し上げる,引き当てる,引当てる,引当る,強いる,強要,当てはまる,当てはめる,当てる,当て嵌まる,当て嵌める,役する,役だてる,役立てる,懸かる,懸る,所望,抛つ,押しつける,押し付ける,押付ける,指示,捧げる,掛ける,掩う,擲つ,施す,施行,求む,求める,活用,無理強,無理強い,用いる,申しこむ,申込む,示す,禀請,稟請,蓋う,蔽う,行う,行ずる,行なう,行使,被う,被せる,要望,要求,覆う,見舞う,連なる,運用,適用,配る,関する,関わる,関係,関連,附ける,頼む,願いでる,願う]

Table 3.6: An example of lexical mapping derived by WordNet.

3.9.3 Lexical Semantics (Pair Relation)

Lexical resources such as WordNet (Miller, 1995) may provide the word alignment links with high precision but low recall using Algorithm 6. This algorithm does not try to specify a particular unique sense, but rather to detect the possible translational equivalences in the sentence pair in question. For IWSLT corpus between JP-EN in Table 3.6, however, despite that we could specify 87% of the possible combinations of content words in monolingual settings, which amounts only to 43.7% of the possible total links. The figure 43.7% is very low compared to GIZA++ of around 70%. For NTCIR-8 patent corpus, despite the high coverage in monolingual settings, this results in a very low percentage of the bilingual correspondence 10.7% for 200k sentence pairs. These figure suggests that the link information derived by WordNet in itself may not be superior to that of word alignment, although this link information may complement the quality. Note that in terms of NPs / MWEs, we will give a statistical approach since the available lexical resources are often too few in number.

Algorithm 6 Algorithm for Lexical Semantic Cue

1. Obtain POS-tag and lemma for each word on the source side.
2. For all lemmas of content words / phrases (NP/VP) in the n-th sentence in the source language, collect all possible synset IDs, and then the derived target words / phrases for each synset ID.
3. Search all possible combinations [13] as to whether each word / phrase in a list contains some word / phrase in the n-th sentence in the target language.
4. If the possibility is unique, the prior probability is assigned to 1. If the possibility is two, the prior probability is assigned to 1/2, and so forth.

size	corpus	unique links	coverage synonym/hype-hypo
50k	NTCIR	8825	79.12 / 81.85
200k	NTCIR	13141	74.52 / 77.80
40k	IWSLT	6808	87.41 / 89.43
115k	literatur	23048	79.16 / 83.49
size	bilingual links	all links	content words
50k	77098	3.8 %	8.5.%
200k	327680	4.2 %	10.7%
40k	68749	16.0 %	43.7%
115k	146584	11.8 %	19.5%

Table 3.7: Statistics related to lexical semantics.

3.9.4 Numbers / Equations (Less Frequent Relation)

There are several classes of linguistic objects which are easy to define monolingually, such as numbers and equations. In this case, 1) we construct a monolingual rule-based extraction algorithm, 2) we extract these objects by a rule-based extraction algorithm respectively, and 3) we perform matching on a bilingual basis. For example, it is fairly easy to write a rule enumerating various possibility that the fragment is numbers, such as figures from 0 to 9 and English word from zero to billion.

3.9.5 Proper Nouns / Transliterations / Localization Terminology (Less Frequent Relation)

Named entity recognition is very sensitive to the trained data which is in-domain data or out-of-domain data in terms of the parallel corpus at hand. For this reason, we do not rely overly on

the results of named-entity recognition, but we first detect such named entities by detecting OOV words. Japanese transliteration is not easy, e.g. Knight and Graehl (Knight and Graehl, 1997) only handle very easy cases. In general, the level of difficulty in transliteration depends on whether there is no fluctuation in Japanese, whether there are less Japanized English involved, whether there is not much transliteration from German or Russian, whether the Japanese way of abbreviation is involved, and whether there is no mixture of 'r' and 'l'.

fluctuation	ロイアリティー,ロイアルティ ロイアルティー,ローヤルティー	royalty, loyalty
fluctuation	ロープウェー,ロープウェイ, ロープウエー,ロープウエイ	ropeway, aerial tram, cable car
fluctuation	ロー付け,ローづけ	soldering, brazing
Japanized way of abbrev.	レフ	reflex camera, reflector
Japanized way of abbrev.	ローマカトリック	Roman Catholic
'e' and 'i'	レバー	lever / joystick,liver
'r' and 'l'	リスト	list, wrist
Japanized English	ロールサンド	roll sandwich, sandwich roll
Japanized English	リストラ策	restructuring scheme
Japanized English	ヨーロッパ連合	European Union, EU
Japanized German	リウマチ	rheumatism (German: Rheumatismus)

Table 3.8: An example of transliteration (Breen, 1999).

equations	103
transliteration	25928
proper nouns	3408
localization	207
symbols	13842

Table 3.9: Statistics of less frequent substructure.

3.10 Experiments

This section describes experiments using some of the algorithms introduced so far. The first experiment is on sentence-level noise reduction. The second experiment is on the MAP-based word

aligner with given NPs as prior knowledge.

3.10.1 Word Alignment with Sentence-level Noise Reduction

Experimental Settings The baseline in our experiments is a standard log-linear PB-SMT system based on Moses. The GIZA++ implementation (Och and Ney, 2003) of IBM Model 4 is used as the baseline for word alignment. Model 4 is incrementally trained by performing 5 iterations of IBM Model 1, 5 iterations of the HMM Model, 5 iterations of IBM Model 3, and 5 iterations of IBM Model 4. For phrase extraction, the grow-diag-final heuristics described in (Koehn et al., 2003) (Refer to Definition 3 and its subsequent explanation) are used to derive the refined alignment from bidirectional alignments. We then perform Minimum Error Rate Training (MERT) (Och, 2003) which optimizes the BLEU metric, while a 5-gram language model is derived with Kneser-Ney smoothing (Kneser and Ney, 1995) trained with SRILM (Stolcke, 2002) on the English side of the training data. We use Moses (Koehn et al., 2007) for decoding.

We evaluate our method using the News Commentary parallel corpus used in the 2007 Statistical Machine Translation Workshop shared task. We use the devset and the evaluation set provided by this workshop. The training set size for EN–ES is 51k and that for DE–EN is 60k.

Experimental Results Table 3.10 shows the results. Although 'noise' does not always correspond to sentences which include many-to-many mapping objects, the improvement achieved by this algorithm was relatively large.

EN–ES	BLEU	effective sent
Base	0.280	99.30 %
Ours	0.317	97.80 %
DE–EN	BLEU	effective sent
Base	0.169	99.10 %
Ours	0.218	97.14 %

Table 3.10: Results for Algorithm 3 (revised good point algorithm).

We conducted experiments in two evaluation campaigns as well. The first one was for IWSLT09

	train set	duplicated train set	redundancies	train set	duplicated train set	noise reduction	removal
BT TR-EN	27,972	20,112	3.0 %	.4831	.4478	.4611	7.1%
BT ZH-EN	47,098	43,657	12.2 %	.3903	.3750	.3741	10.4%
CH ZH-EN	75,231	69,680	4.0 %	.3169	.2847	.3011	10.6%
CH EN-ZH	39,228	38,227	12.0 %	.3531	.3154	.3170	9.5%

Table 3.11: Redundancies in Parallel corpus and its BLEU score improvement. BT denotes BTEC corpus while CH denotes Challenge corpus. TR is an abbreviation of Turkish, while ZH is that of a simplified Chinese.

(Ma et al., 2009) and the second one was for NTCIR-8 (Okita et al., 2010b). One important finding in IWSLT09 was that Algorithm 3 (revised good point algorithm) did not work well at all. In Table 3.11, the column labelled with 'train set' shows the BLEU scores for original training set, the column labelled with 'pure train set' shows BLEU scores for the pure training set without redundant sentences, and the column labelled with 'noise reduction' shows the BLEU scores for the reduced training set. In sum, we conjecture that due to a lot of duplicated sentence pairs in the IWSLT09 data sets, Algorithm 3 did not work so well. Although we tuned parameters empirically, the sentence duplication algorithm works comparably well to the noise reduction algorithm (Okita, 2009b) (Refer to Algorithm 4).

In NTCIR-8 parallel corpus between EN–JP, we conduct the method for 600k sentence pairs (Okita et al., 2010b). The left half of Table 3.12 shows the result for EN–JP. HPB-SMT 1 is Moses and HPB-SMT 2 is Joshua (Li et al., 2009). PB-SMT 1 is Moses with the distortion limit 12 over a 600k training corpus, while PB-SMT 2 is Moses with the distortion limit 6 over a 3,200k training corpus. It is noted that the official BLEU scores containing an asterisk are evaluated after the removal of Out-Of-Vocabulary (OOV) words. It is noted that we trained over 3,200k training corpus for the systems marked with $^{+}$ and over 600k training corpus for other systems.

The right half of Table 3.12 shows the result for JP–EN. The HPB-SMT 1 is based on the Moses Chart, and the HPB-SMT 2 is based on Joshua. The PB-SMT 1 system is based on Moses with the distortion limit 12 over 600k training corpus, while PB-SMT 2 is based on Moses with

the distortion limit 6 over 3,200k training corpus. It is noted that we trained over 3,200k sentence pairs for the systems marked with $^{+}$ and used over 600k training data for other systems.

Systems (JP–EN)	BLEU	#OOV	Systems (EN–JP)	BLEU
HPB-SMT 1	26.86*	314	HPB-SMT 1	32.50
PB-SMT 1	26.51*	194	PB-SMT 1	30.53
Noise reduction (PB-SMT)	24.01	443	PB-SMT 2^{+}	30.08
PB-SMT 2^{+}	23.91*	316	Noise reduction	29.53
HPB-SMT 2	23.30	303	HPB-SMT 2	27.23

Table 3.12: Intrinsic evaluation results (JP–EN and EN–JP).

3.10.2 MAP-based Word Aligner with Noun Phrase Detection

This setting is to verify the effectiveness of our word-level noise-sensitive MAP-based word aligner presented in Section 3.8.3. The extraction of NPs (or MWEs) is described in Section 3.9.5 of NPs / MWEs /Idioms.

Experimental Settings The baseline in our experiments is a standard log-linear phrase-based MT system based on Moses. The GIZA++ implementation (Och and Ney, 2003) of IBM Model 4 is used as the baseline for word alignment, which we compare to our modified GIZA++ (Section 3.4.3). Model 4 is incrementally trained by performing 5 iterations of Model 1, 5 iterations of HMM, 5 iterations of Model 3, and 5 iterations of Model 4. For phrase extraction the grow-diag-final heuristics (Koehn et al., 2003) are used to derive the refined alignment from bidirectional alignments. We then perform MERT (Och, 2003) while a 5-gram language model is trained with SRILM (Stolcke, 2002). Our implementation is based on a modified version of GIZA++ (Och and Ney, 2003). We modify the function that reads a bilingual terminology file, the function that calculates priors, the M-step in IBM Models 1-5, and the forward-backward algorithm in the HMM Model. Other related software tools are written in Python and Perl: terminology concatenation, terminology numbering, and so forth.

We use two corpora: the NTCIR-8 corpus (Fujii et al., 2010) with heuristic-based and statistical

noun phrase extraction and Europarl (Koehn, 2005) with statistical noun phrase extraction.

corpus	language	size	#unique NPs	#all NPs
statistical method				
NTCIR	EN-JP	200k	1,121	120,070
europarl	EN-FR	200k	312	22,001
europarl	EN-ES	200k	406	16,350
heuristic method				
NTCIR	EN-JP	200k	50,613	114,373

Table 3.13: Statistics of our noun phrase extraction method. The numbers of noun phrases are from 0.08 to 0.6 NP / sentence pair in our statistical noun phrase extraction methods.

Experimental Results Firstly, noun phrases are extracted from both corpora. In the second step, we apply our modified version of GIZA++ in which we incorporate the results of noun phrase extraction. Secondly, in order to incorporate the extracted noun phrases, they are reformatted as shown in Table 3.13. Thirdly, we convert all noun phrases into a single token, i.e. we concatenate them with an underscore character. We then run the modified version of GIZA++ and obtain a phrase and reordering table. In the fourth step, we split the concatenated noun phrases embedded in the third step. Finally, in the fifth step, we run MERT, and proceed with decoding before automatically evaluating the translations.

size	EN-JP	BLEU	JP-EN	BLEU
50k	baseline	16.33	baseline	22.01
50k	baseline2	16.10	baseline2	21.71
50k	prior	17.08	prior	22.11
200k	baseline	23.42	baseline	21.68
200k	baseline2	24.10	baseline2	22.32
200k	prior	24.22	prior	22.45

Table 3.14: Results for EN-JP. Baseline is plain GIZA++ / Moses (without NP grouping / prior), baseline2 is with NP grouping, prior is with NP grouping and prior.

Table 3.14 and 3.15 show the results where ‘baseline’ indicates no Bilingual NP (BNP) group-

size	FR-EN	BLEU	EN-FR	BLEU
50k	baseline	17.68	baseline	17.80
50k	baseline2	17.76	baseline2	18.00
50k	prior	17.81	prior	18.02
200k	baseline	18.40	baseline	18.20
200k	baseline2	18.80	baseline2	18.50
200k	prior	18.99	prior	18.60
size	ES-EN	BLEU	EN-ES	BLEU
50k	baseline	16.21	baseline	15.17
50k	baseline2	16.61	baseline2	15.60
50k	prior	16.91	prior	15.87
200k	baseline	16.87	baseline	17.62
200k	baseline2	17.40	baseline2	18.21
200k	prior	17.50	prior	18.20

Table 3.15: Results for FR-EN and ES-EN. Baseline is plain GIZA++ / Moses (without bilingual noun phrase grouping / prior), baseline2 is with bilingual noun phrase grouping, prior is with bilingual noun phrase grouping and prior.

ing nor any prior, and 'baseline2' represents a BNP grouping but without the prior. Although 'baseline2' (BNP grouping) shows a drop in performance in the JP–EN / EN–JP 50k sentence pair setting, Prior Model results in an increase in performance in the same setting. Except for EN–ES 200k, our Prior Model was better than 'baseline2' and statistically significant. For EN–JP NTCIR using 200k sentence pairs, we obtained an absolute improvement of 0.77 BLEU points compared to the 'baseline'; for EN–JP using 50k sentence pairs, 0.75 BLEU points; and for ES–EN Europarl corpus using 200k sentence pairs, 0.63 BLEU points.

For EN–JP NTCIR using the same corpus of 200k, although the number of unique NPs extracted by the statistical method and the heuristic method varies significantly, the total number of NPs extracted by each method becomes comparable. The resulting BLEU score for the heuristic method (24.24 / 22.48 Blue points for 200k EN–JP / JP–EN) is slightly better than that of the statistical method. The possible reason for this is related to the way the heuristic method groups terms including reference numbers, while the statistical method does not. As a result, the complexity of the alignment models simplified slightly in the case of the heuristic method.

3.11 Conclusions

We presented two methods in this capter. One is for sentence-level noise (Section 3.8) and the other is for word-level noise (Sections 3.5, 3.6, 3.7 and 3.9). For the sentence-level noise we employed the method similar to the outlier detection algorithm, while for the word-level noise we built a MAP-based word aligner.

The sentence-level noise reduction was easy to implement, but was difficult to handle. We observed that the sentence-level noise reduction worked for the small dataset, but was not so successful for the redundant dataset and the large dataset. Except for the case where we obtain good performance, we will be lost when we faced with the situation where we find that the noise reduction does not work. This is because there is little space for us to control this algorithm. In this sense, this algorithm has a disadvantage in that we will be lost when this algorithm does not work. All the more, it seems that whether this algorithm works or not depends on the dataset. This is because the success of this algorithm depends on the success of word alignment. Obviously it is quite difficult to predict whether the given parallel corpus is easier to align or not compared to other parallel corpus. Similarly, it is difficult to predict whether the given redundancies in parallel corpus affect the performance of word alignment.

The word-level noise reduction requires a lot of preparations, and is difficult to predict the result based on the supplying prior knowledge about word alignment links. The preparation includes a construction of MAP-based word aligner as well as the investigation how to detect word alignment links through various linguistic knowledge where each detection tend to require different methods. Our experiments in this thesis have scratched only the surface of it and we left many things as a further study. Our findings are that if we supply word alignment links of many-to-many mapping objects such as NPs, this resulted in the encouraging improvement of 0.63 - 0.77 BLEU points absolute. For translational noise (which does not need a MAP-based word aligner objects though), we observed in PB-SMT that only removing five typical Japanese noisy phrases improved 0.75 - 1.00 BLEU points absolute. One issue we noticed was that the smoothing of translation model

may improve the performance. We can guess that the way that the MAP-based word aligner is constructed will make the distribution radically change compared to the traditional way of making a translation model without any prior knowledge. We will handle this topic in Chapter 5.

Chapter 4

Smoothing Methods: Overfitting

Statistical approaches or non-parametric Machine Learning methods estimate some targeted statistical quantities based on (i) the (true) posterior distributions in a Bayesian manner (Bishop, 2006) or (ii) on the underlying fixed but unknown (joint) distributions from which we assume that we sample our training examples in a frequentist manner (Vapnik, 1998). In Natural Language Processing (NLP), such distributions are observed by simply counting (joint/conditional) events, such as $c(w)$, $c(w_1, w_2)$ and $c(w_3|w_1, w_2)$ where w denotes words and $c(\cdot)$ denotes a function to count events. Since such quantities are often discrete, it is unlikely that such events will be counted incorrectly at first sight. However, it is a well-known fact in NLP that such counting methods are often unreliable if the size of the corpus is too small compared to the model complexity. Researchers in NLP often try to rectify such counting of (joint or conditional) events using a technique known as smoothing (Kneser and Ney, 1995). Most smoothing techniques do not have a statistical model but rely on either interpolation or back-off schemes.

Chapter 4 discusses a statistical smoothing method based on (hierarchical) Pitman-Yor processes, which is a nonparametric generalization of the Dirichlet distribution that produces power-law distributions (Teh, 2006; Goldwater et al., 2006). Various pieces of research have been carried out in which hierarchical Pitman-Yor processes have been applied to language models (Hierarchi-

cal Pitman-Yor Language Model (HPYLM) (Teh, 2006; Mochihashi and Sumita, 2007; Huang and Renals, 2009)) whose generative model uses hierarchies of n-grams. This model is shown to be superior to the interpolated Kneser-Ney methods (Kneser and Ney, 1995) and comparable to the modified Kneser-Ney methods in terms of perplexity. Although this method was presented five years ago, there has been no paper which reports that this language model indeed improves translation quality in the context of Machine Translation (MT). This is important for the MT community since an improvement in perplexity does not always lead to an improvement in BLEU score; for example, the success of word alignment measured by Alignment Error Rate (AER) does not often lead to an improvement in BLEU (Fraser and Marcu, 2007).

Section 4.1 describes the smoothing methods used for language models: a Hierarchical Pitman-Yor Language Model (HPYLM) and a Pitman-Yor Good-Turing Language Model (PYGTLM). The primary aim is to report in the context of MT that an improvement in perplexity really leads to an improvement in BLEU score. It turns out that an application of HPYLM requires a minor change in the conventional decoding process. We conducted experiments in which HPYLM improved translation by 1.03 BLEU points absolute and 6% relative for 50k EN–JP, which was statistically significant (Koehn, 2004). Unfortunately, our experimental results show that PYGTLM performs better than the conventional Kneser-Ney method and Good-Turing method, but is inferior to HPYLM.

Section 4.2 proposes a new Pitman-Yor process-based applied to translation models. In fact, this section is the real motivation for this chapter. In Chapter 5, we proposed a new word alignment method which incorporates a range of linguistic knowledge. The first hypothesis is that even if we build a translation model without such additional linguistic knowledge, the relative frequency estimates of the translation model will not be correctly calculated, as was discussed by Foster et al. (Foster et al., 2006). The second hypothesis is that if we incorporate such additional linguistic knowledge in the word alignment process, the relative frequency estimates of the translation model will include further irregularities. Hence, in both cases, the application of smoothing meth-

ods is considered to improve the results. Under such conjectures, we propose a new smoothing method, which is a straightforward extension of the HPYLM in Section 4.1. The result shows our hypotheses to be well-founded.

4.1 Language Model Smoothing

4.1.1 Language Model

We start with several traditional smoothing methods for language models in order to understand that these are really the methods which combine some heuristics (Manning and Schutze, 1999; Jurafsky and Martin, 2009; Koehn, 2010). Such heuristics include absolute discount, back-off, interpolation schemes, and so forth. The main motivation behind our focus on the hierarchical Pitman-Yor process-based smoothing method in this chapter lies in the fact that we would like to base our models on more statistically motivated methods rather than those methods explained in this section.

In Section 4.1.1, we will informally explain the difference of smoothing methods using the bigram language model. This will not lose generality, but will make it easier to understand. Before we discuss these methods, we introduce some notation: $w_{i-1}w_i$ denotes two consecutive words, $\cdot w$ denotes the consecutive two words where the first word is any word, and $c(\cdot)$ denotes a function which counts the words specified as its argument. The condition $c(w_{i-1}w_i) > 0$ means that the bigram $w_{i-1}w_i$ appeared in the corpus. [1]

We start with a maximum likelihood method, which is shown in (4.1).

$$P_{ML}(w_i|w_{i-1}) = \begin{cases} \dfrac{c(w_{i-1}w_i)}{\sum_w c(w_{i-1}w)} & \text{if} \quad c(w_{i-1}w) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

Since the maximum likelihood reflects purely statistics, there is no value assigned for unobserved

[1] Hence, in most cases below, the condition 'otherwise' means that the bigram $w_{i-1}w_i$ did not appear in the corpus.

n-grams, which is shown in otherwise in (4.1). If we subtract a fixed (absolute) discount D from each count in order to allocate some mass for unobserved bigrams, this is called the absolute discounting method, which is shown in (4.2):

$$P_{AbsoluteDiscounting}(w_i|w_{i-1}) = \begin{cases} \dfrac{c(w_{i-1}w_i) - D}{\sum_w c(w_{i-1}w)} & \text{if} \quad c(w_{i-1}w) > 0 \\ \alpha(w_i)p(w_i) & \text{otherwise} \end{cases} \tag{4.2}$$

If we take into account the diversity of histories for the unobserved bigrams, this is called a Kneser-Ney smoothing method (Kneser and Ney, 1995). With the definition of the count of histories for a word as in (4.3),

$$N_{1+}(\bullet w) = |\{w_i : c(w_{i-1}w) > 0\}| \tag{4.3}$$

the raw counts of the maximum likelihood estimation are replaced with this count of histories for a word. In sum, a Kneser-Ney method is written as in (4.4):

$$P_{KneserNey}(w_i|w_{i-1}) = \begin{cases} \dfrac{c(w_{i-1}w_i) - D}{\sum_w c(w_{i-1}w)} & \text{if} \quad c(w_{i-1}w) > 0 \\ \alpha(w_i)\dfrac{N_{1+}(\bullet w)}{N_{1+}(w_{i-1}w)} & \text{otherwise} \end{cases} \tag{4.4}$$

If we combine the ideas behind interpolation and back-off, we can combine two terms in the right-hand side in (4.4). This is called an **interpolated** Kneser-Ney method (Chen and Goodman, 1998), which is shown in (4.5):

$$P_{InterpolatedKN}(w_i|w_{i-1}) = \begin{cases} \dfrac{c(w_{i-1}w_i) - D}{\sum_w c(w_{i-1}w)} + \beta(w_i)\dfrac{N_{1+}(\bullet w)}{N_{1+}(w_{i-1}w)} & \text{if} \quad c(w_{i-1}w) > 0 \\ \beta(w_i)\dfrac{N_{1+}(\bullet w)}{N_{1+}(w_{i-1}w)} & \text{otherwise} \end{cases} \tag{4.5}$$

Now, if we have an intuition that an absolute discount D_n for each n-gram takes different (but

fixed) values shown in (4.6),

$$D(n) = \begin{cases} D_1 & (\text{if} \quad c = 1) \\ D_2 & (\text{if} \quad c = 2) \\ D_{3+} & (\text{if} \quad c \geq 3), \end{cases} \tag{4.6}$$

this method makes a **modified** Kneser-Ney method (Chen and Goodman, 1998), which is shown in (4.7). Note that we derive (4.7) from (4.5) using (4.6) for different sized n-grams. Note that similarly with (4.6), although each distribution has the case when a bigram is not observed it is omitted from (4.7).

$$\begin{cases} P_{ModifiedKN}(w_i) & = & \frac{c(w_i) - D_1}{\sum_w c(w)} + \beta(w_i)\frac{N_{1+}(\bullet)}{N_{1+}(w)} \\ & & (\text{if W=unigram}) \\ P_{ModifiedKN}(w_i|w_{i-1}) & = & \frac{c(w_{i-1}w_i) - D_2}{\sum_w c(w_{i-1}w)} + \beta(w_i)\frac{N_{1+}(\bullet w)}{N_{1+}(w_{i-1}w)} \\ & & (\text{if W=bigram}) \\ P_{ModifiedKN}(w_i|w_{i-2}w_{i-1}) & = & \frac{c(w_{i-2}w_{i-1}w_i) - D_{3+}}{\sum_w c(w_{i-2}w_{i-1}w)} + \beta(w_i)\frac{N_{1+}(\bullet w_{i-1}w)}{N_{1+}(w_{i-2}w_{i-1}w)} \\ & & (\text{if W} \geq \text{trigram}) \end{cases} \tag{4.7}$$

A Good-Turing method (Good, 1953) introduces the count-of-counts N_c shown in (4.8),

$$N_c = \sum_{x:count(x)=c} 1, \tag{4.8}$$

which is the number of different words that were seen exactly c times. Using this N_c, this method infers the zero probability mass. Let N denote the total number of counts. The modified count c^*

can be obtained by

$$c^* = (c+1)\frac{N_{c+1}}{N_c} \tag{4.9}$$

Using these quantities, the probability mass for unobserved n-grams can be calculated as in (4.10) where the mass for unobserved n-grams is uniformly allocated:

$$P_{GoodTuring}(w_1,\ldots,w_n) = \begin{cases} \dfrac{c^*}{N} & \text{if} \quad c(w_1,\ldots,w_n) > 0 \\ \dfrac{1-\sum_{i=1}^{\infty} c^* \dfrac{N_i}{N}}{N_0} & \text{if} \quad c(w_1,\ldots,w_n) = 0 \end{cases} \tag{4.10}$$

4.1.2 Hierarchical Pitman-Yor Process-based Language Model

According to Teh, the performance of a hierarchical Pitman-Yor is known to be comparable with the modified Kneser-Ney smoothing in terms of perplexity (Teh, 2006). As was shown in Section 4.1.1, the modified Kneser-Ney method employs several heuristics such as interpolation, absolute discount, and back-off. Compared to this, a hierarchical Pitman-Yor process-based smoothing is mathematically constructed using Bayesian statistics.

HPYLM: Generative Model A Hierarchical Pitman-Yor Language Model (HPYLM) (Goldwater et al., 2006; Teh, 2006; Mochihashi et al., 2009; Okita and Way, 2010) is constructed encoding the property of the power-law distribution.

Let $PY(d,\theta,G_0)$ denote a Pitman-Yor process (Pitman, 1995), d denote a discount parameter, θ denote a strength parameter, and G_0 a base distribution. We define $\pi(u)$ as the suffix of u consisting of all but the earliest word in Equation (4.11), as in (Teh, 2006): we see u as n-gram words and $\pi(u)$ as (n-1)-gram words. Then, we place a Pitman-Yor process prior *recursively* over

$G_{\pi(u)}$ in the generative model, as is shown in (4.11):

$$\begin{cases} G_u|d_{|u|}, \theta_{|u|}, G_{\pi(u)} \sim PY(d_{|u|}, \theta_{|u|}, G_{\pi(u)}) \\ \dots \\ G_\emptyset|d_0, \theta_0, G_0 \sim PY(d_0, \theta_0, G_0) \end{cases} \tag{4.11}$$

Note that the discount and strength parameters are functions of the length $|u|$ of the context, while the mean vector is $G_{\pi(u)}$, and the vector of probabilities of the current word given all but the earliest word in the context.

HPYLM: Inference One procedure — A Chinese restaurant process — computes inference in order to generate words drawn from G, which iteratively marginalizes out G.

Let h be an n-gram context; for example in trigrams, this is $h = \{w_1, w_2\}$. A Chinese restaurant contains an infinite number of tables t, each with infinite seating capacity. Customers, which are the n-gram counts $c(w|h)$, enter the restaurant and seat themselves over the tables $1, \dots, t_{hw\cdot}$.[2] The first customer sits at the first available table, while each of the subsequent customers sits at an occupied table with probability proportional to the number of customers already sitting there $c_{hwk} - d$, or at a new unoccupied table with probability proportional to $\theta + d \cdot t_{h\cdot}$ as is shown in (4.12):

$$w|h \sim \begin{cases} c_{hwk} - d & (1 \leq k \leq t_{hw}). \\ \theta + d \cdot t_{h\cdot} & (k = new). \end{cases} \tag{4.12}$$

where c_{hwk} is the number of customers seated at table k until now, and $t_{h\cdot} = \sum_w t_{hw}$ is the total number of tables in h.

Hence, the predictive distribution of n-gram probability in HPYLM is recursively calculated

[2] In $t_{hw\cdot}$, t_X means the table for the word sequence X. $hw\cdot$ means the sequence hw followed by some word.

as in (4.13):

$$p(w|h) = \frac{c(w|h) - d \cdot t_{hw}}{\theta + c(h)} + \frac{\theta + d \cdot t_{h\cdot}}{\theta + c(h)} p(w|h') \tag{4.13}$$

where $p(w|h')$ is the same probability using a $(n$-1)-gram context h'. Implementation of this inference procedure relates to the Markov chain Monte Carlo sampling (Metropolis et al., 1953). The simplest way is to build a Gibbs sampler (Geman and Geman, 1984) while a more efficient way is to build a blocked Gibbs sampler (Mochihashi et al., 2009).

Decoding Algorithm in PB-SMT A minor difference in the decoding process is required. In a test sentence, if we encounter unseen phrases, a conventional PB-SMT decoder looks up the probability with constant zero-probabilities. However, our algorithm should look up the corresponding probabilities based on the hierarchical Pitman-Yor processes. We calculate these zero-probabilities using the parameters that we derived while obtaining the HPYLM.

There are two ways to incorporate this: 1) just before we do decoding, we update a language model by supplying a test sentence in terms of zero-probabilities (that is, if test sentences include unseen words and phrases, we notify the translation model to incorporate these unseen words and phrases.), and 2) we modify a PB-SMT decoder to incorporate this difference. Due to its easiness of implementation,[3] we take the approach 1) here, but the effect would be the same.

Our procedure is as follows. Firstly, we prepare the HPYLM parameter file $p_0(w)$ which we obtained when we calculate the HPYLM. This HPYLM parameter file contains the parameters in the Chinese restaurant processes, such as the number of tables, d, θ, and so forth. Such parameters enable us to calculate the zero-probabilities for any unseen phrases in a test sentence. The overall algorithm to obtain the updated HPYLM is shown in Algorithm 7.

[3]If we modify a PB-SMT decoder, we need to modify the stack decoding algorithm. This will take time.

Algorithm 7 Decoder for HPYLM $p(w)$

Given: a test sentence $\breve{s} = \{\breve{s}_1, \cdots, \breve{s}_n\}$, HPYLM $p(w)$, HPYLM parameter file $p_0(w)$.
Step 1: By generating a possible n-gram candidate, using p_0 we update HPYLM $p'(w)$.
Step 2: Run a decoder which looks up updated HPYLM $p'(w)$.

4.1.3 Good-Turing Pitman-Yor Language Model Smoothing

We use the same generative model which uses the Pitman-Yor process as a prior in Equation (4.11) once (not recursively), and let us now consider a count-counts function. (This is also known as event-counts or count of counts.) We refer to this model as a Good-Turing Pitman-Yor Language Model (GTPYLM). Our intention here is to incorporate the prior knowledge that a distribution takes a power-law distribution, as well as incorporating the zero-frequency mass.

We use the notation of (4.8) and (4.10). By (4.13), the predictive distribution of n-gram probability in GTPYLM is computed as in (4.14):

$$p^*(w_i|w_i-1, N_w) = \frac{c^*(w|w_{i-1}, N_w) - d \cdot t_{N_w w}}{\theta + c^*(w|N_w)} + \frac{\theta + d \cdot t_{N_w \cdot}}{\theta + c^*(w|N_w)} p^*(w_i|w_{i-1}, N_{w-1}) \quad (4.14)$$

Note that this formulation does not avoid the problem of data sparseness of N_c when c is large, which requires us to obtain N_c in a similar way as in other work, such as Gale (1994).

4.1.4 Performance

We conduct an experimental evaluation for JP–EN on the NTCIR-8 corpus (Fujii et al., 2010) and for FR–EN and ES–EN on Europarl (Koehn, 2005). We randomly extracted a training corpus of 200k sentence pairs where we use 1,200 sentence pairs (NTCIR) and 2,000 sentence pairs (Europarl) for the development set, and we use 1,119 (EN–JP) / 1,251 (JP–EN) sentence pairs (NTCIR) and 2,000 sentence pairs (Europarl; test2006) for our test set.[4]

Our baseline was a standard log-linear PB-SMT system based on Moses (Koehn et al., 2007). The GIZA++ implementation (Och and Ney, 2003) of IBM Model 4 was used for word alignment.

[4]The number of 1,119 and 1,251 are provided by NTCIR-8 organizers.

For phrase extraction the grow-diag-final heuristics described in (Och and Ney, 2003) was used to derive the refined alignment. We then performed MERT (Och, 2003) to optimize the BLEU metric, while a 5-gram language model was derived with Kneser-Ney smoothing trained with SRILM (Stolcke, 2002) on the English side of the training data. We used Moses for decoding.

size	EN–JP	BLEU	Perplexity	JP–EN	BLEU	Perplexity
200k	baseline1	23.42	59.607	baseline1	21.68	117.78
200k	baseline2	23.36	58.587	baseline2	21.38	119.13
200k	HPYLM	24.22	52.295	HPYLM	22.32	105.22
200k	GTPYLM	23.22	53.332	GTPYLM	22.21	110.12
size	FR–EN	BLEU	Perplexity	EN–FR	BLEU	Perplexity
200k	baseline1	18.40	162.573	baseline1	18.20	165.839
200k	baseline2	18.19	165.232	baseline2	18.02	168.989
200k	HPYLM	18.99	148.338	HPYLM	18.60	153.921
200k	GTPYLM	18.70	152.104	GTPYLM	18.50	160.332
size	ES–EN	BLEU	Perplexity	EN–ES	BLEU	Perplexity
200k	baseline1	16.87	168.431	baseline1	17.62	154.273
200k	baseline2	16.37	174.856	baseline2	17.32	168.754
200k	HPYLM	17.50	152.312	HPYLM	18.20	145.223
200k	GTPYLM	17.15	156.440	GTPYLM	18.10	146.211

Table 4.1: Results for language model. Baseline1 uses modified Kneser-Ney smoothing and baseline2 uses Good-Turing smoothing.

This method conjectures that the weakness of language modelling and translation modelling is its ignorance of underfitting in terms of the number of samples. HPYLM algorithm implements a statistical smoothing method which deals with this underfitting problem. The results show that a hierarchical Pitman-Yor process-based language model indeed improves the translation quality, which is shown in Table 4.1. The best improvement of HPYLM with regard to the modified Kneser-Ney method was 0.80 BLEU points absolute and 3.4 % relative for 200k EN–JP corpus, while the best improvement of GTPYLM with regard to the Good-Turing method was 0.83 BLEU points absolute and 3.9 % relative for 200k JP–EN corpus.

4.2 Translation Model Smoothing

When PB-SMT was introduced around 2002, the most primitive method, namely relative frequency (Koehn et al., 2003), was introduced to calculate the probabilities in the translation model. Then, two kinds of glass box smoothing methods were introduced by (Zens and Ney, 2004) and (Koehn et al., 2005), which decompose source phrases by independence assumptions. Foster et al. apply to a translation model the classical smoothing methods used in language models, such as the Good-Turing method and the Kneser-Ney method (Foster et al., 2006). Johnson et al. show that the performance does not decreased much even if most of the phrases are pruned (Johnson et al., 2007).

Since most of these methods involve a combination of heuristics, one motivation here is to seek the most well-founded method for the translation model. However, as is shown by Foster et al., the modification of smoothing methods used for language model to the case of translation models is quite straightforward. We consider to apply a hierarchical Pitman-Yor process-based smoothing method to the translation model, which is the theme of this section.

4.2.1 Hierarchical Pitman-Yor Translation Model Smoothing

An n-gram is often defined as a subsequence of n items from a given sequence where items can be phonemes, syllables, letters, words or base pairs. Although we can extend this definition of n-gram to one which includes 'phrases', let us use the different term 'n-phrase-gram' instead in this chapter, in order to avoid any confusion with the n-gram for words. Fig. 4.1 shows a typical phrase extraction example. In this process, under the consistency constraint, phrase pairs are extracted which is depicted in the centre. Note that this figure is depicted separating the source and the target sides.

Fig. 4.2 shows the same figure if we depict elements as pairs. The lowest column includes only 1-phrase-grams, the second lowest column includes 2-phrase-grams, and so on. The line connecting two nodes indicates parent-child relations. Accordingly, this becomes the lattice structure of

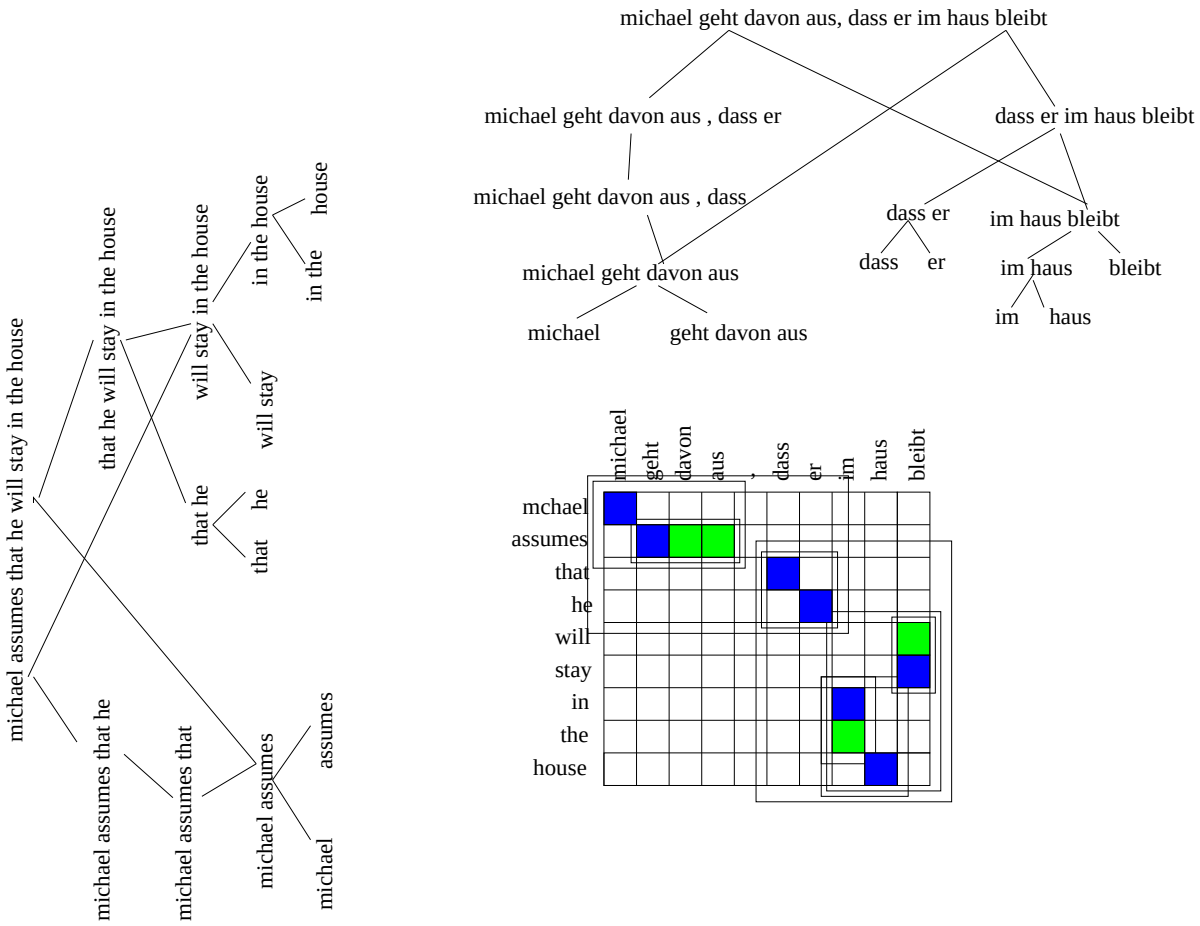

Figure 4.1: A toy example of phrase extraction process. Resultant phrase pairs can be described as a lattice structure.

the generated phrase pairs. These generated phrase pairs may have several paths to yield complete sentences. Similar to the HPYLM case, we can limit this by considering the suffix of a sequence, meaning that we can process a sequence always from left-to-right. Hence, although the natural lattice would include the dashed lines, the dashed lines can be eliminated if we impose the constraint that we should always read the suffix of this sequence from left-to-right. This constraint makes the resulting structure a tree. If the resulting structure is a tree, we can employ the same strategy as we did with HPYLM. The predictive distribution can be calculated by Equation (4.13) by replacing n-grams with n-phrase-grams.

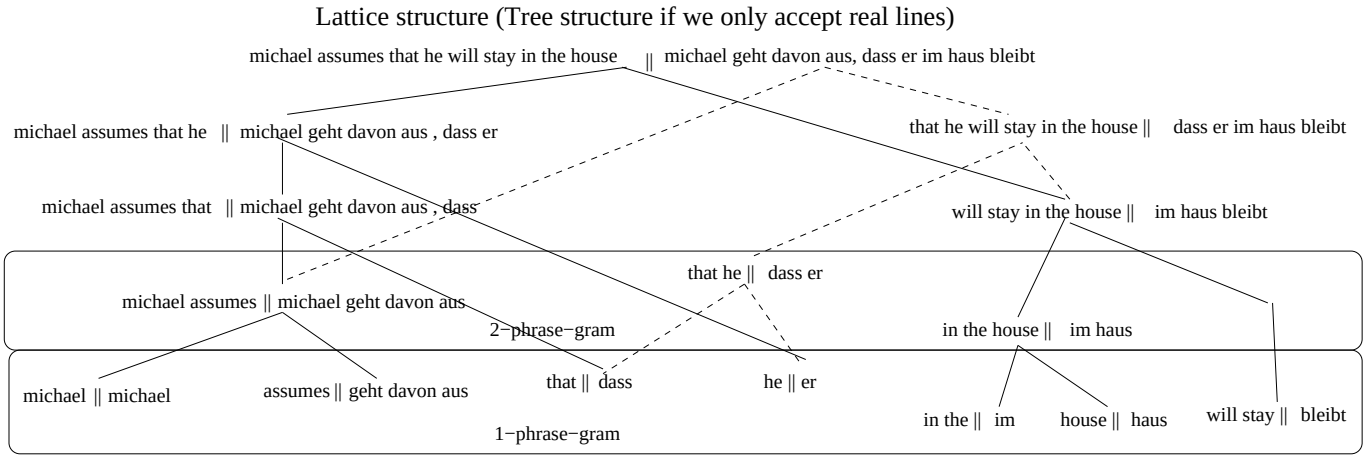

Figure 4.2: Figure shows a lattice structure of translation model for a toy example.

4.2.2 Performance

We randomly selected 200k sentence pairs from the NTCIR-8 patent corpus for JP–EN (Fujii et al., 2010) as a training corpus. We used 1.2k sentences for the development set, while we used the test set prepared for the NTCIR-8 evaluation campaign. The Japanese side of the data was segmented using Cabocha (Kudo and Matsumoto, 2003). Table 4.2 shows the statistics of each type of prior knowledge. We prepared terminology without using external resources but with some human interaction. For the first prior knowledge type, NPs were extracted by the heuristic NP-extraction strategy similar to (Kupiec, 1993), and then the extracted terminology was processed by hand via manual inspection of the data. For the second prior knowledge type, paraphrases were extracted by the method described in (Bannard and Callison-Burch, 2005). For the third prior knowledge type, OOV word lists were created as follows. We constructed a PB-SMT decoder, decoded all the training corpus as well as the test corpus, and collected all of the OOV words from the translation outputs. Then, we supplied the translation counterparts manually. [5]

Table 4.3 shows our results. Without translation model smoothing, the improvement in BLEU by the prior 1 was 0.80 BLEU points absolute, the prior 2 was 0.65 BLEU points absolute, the

[5]Due to the segmentation process, for around 20% of the transliteration terms it was not possible to find their counterparts.

JP–EN		training	test
prior knowledge 1	NPs	120070	3865
prior knowledge 2	paraphrases	432135	—
prior knowledge 3	transliteration	25928	284
	proper nouns	3408	2
	localization	207	2
	equations	103	1
	symbols	13842	684
	noise	19007	175

Table 4.2: Statistics of prior knowledge.

JP–EN	without TM smoothing[6]	with TM smoothing
baseline	21.68	22.44
prior 1	22.48	22.78
prior 2	22.43	22.64
prior 3	22.26	22.52
all 1-3	22.95	23.03
heuristics	21.90	22.49

Table 4.3: Results for 200k JP–EN sentences. Heuristics in the last row shows the result when prior knowledge 1 was added at the bottom of the translation model.

prior 3 was 0.58 BLEU points absolute, and the prior 1 to 3 was 1.27 BLEU points absolute. With translation model smoothing, the improvement in BLEU compared to the baseline with no TM smoothing by the prior 1 was 1.10 BLEU points absolute, the prior 2 was 0.96 BLEU points absolute, the prior 3 was 0.85 BLEU points absolute, and the prior 1 to 3 was 1.35 BLEU points absolute. With translation model smoothing, the improvement in BLEU compared to the baseline with TM smoothing by the prior 1, 2 and 3 was rather small. This shows that an MWE-sensitive aligner and the translation model smoothing improved the results if we applied them separately, but the combined effect was not much observed unless we incorporate NPs, MWEs, paraphrases, and OOVs together.

[6]The probability in a translation model is described by relative frequency.

4.3 Conclusions

This chapter presents an application of the (hierarchical) Pitman-Yor process-based language model and translation model to MT. The first part discusses language models. Firstly, although the performance of HPYLM was reported in terms of perplexity, there have been no reports, as far as we know, in terms of BLEU in the MT context. We showed that there was a gain with a minor change in the decoding process. Although Teh reported that HPYLM showed a comparable performance with the modified Kneser-Ney method, we obtained better results than the modified Kneser-Ney method here. Secondly, we proposed an alternative language model using the Pitman-Yor process applying the count-counts distribution of the Good-Turing method. The performance of this was not as successful as HPYLM, but it was better than both the modified Kneser-Ney and Good-Turing methods. Furthermore, this was statistically significant.

The second part discusses a translation model. The application of this was a straightforward application of HPYLM introducing a phrase-gram. We conduct experiments combining several types of linguistic knowledge that we obtained in the word alignment process, such as NPs, MWEs, paraphrases, and OOV items. We assume that this method is used for smoothing the probabilities extracted by the MWE-sensitive word alignment method. When we only use the HPYTM, our results show that the improvement was 0.56 BLEU points absolute, but when we incorporate all sorts of linguistic knowledge, we obtained an improvement of 1.35 BLEU points absolute and 6.0% relative.

Chapter 5

Conclusions and Further Study

This thesis examined word alignment and language modelling, exploring 1) the existence of *noise* in word alignment, 2) the existence of *prior knowledge* for word alignment, and 3) the existence of underfitting in terms of the number of samples for language modelling and translation modelling. The first two relate to word alignment in Section 3 and the third one relates to statistical smoothing in Section 4.

In word alignment, this thesis considers not only algorithmic aspects but also data manipulation aspects. The latter is often not considered especially in terms of noise. Our algorithms aim at extracting possible structured objects separately from the word alignment algorithm, then supplying them as a prior knowledge in the word alignment. As is similar with the phrase extraction heuristics (Och and Ney, 2003) which radically reduces the computational complexity compared to the phrase alignment approach of Marcu and Wong (2002), this way of handling structures in word alignment has merits in terms of computational complexity. In order to continue in this way, we provided the MAP-based word alignment technique which was the key component in our approach theoretically speaking.

In practice, we consider two distinct applications: sentence-level and word-level noise reduction. The latter is far more complex compared to the former. The first application handled sentence-

level noise. If we detect such noise, we filtered out such noisy training sentences. Although this was a simple experiment, this study gave us a view as to how we can tackle the problem of noise. If the parallel corpus is not noisy, there are not many things that we can do, as observed in the EN-ZH IWSLT corpus. However, if the corpus is noisy, the effect of our noise reduction method may be significant.

The second and the third applications handle word-level noise: many-to-many mapping objects and translational noise. Despite the fact that we are not given information about alignment links, NP detection gave such information virtually 'for free'. Using such prior knowledge and using a MAP-based word aligner, we incorporate NPs into the word alignment process. We noted that many-to-many mapping objects pose two different challenges for word alignment—noise (or outliers), as well as valid training data—so this situation is not just an application of outlier detection in pattern analysis. We also noted that since not all the many-to-many mapping objects are obstacles for the word alignment process, methods which filter out any kind of noise will often give better results than those which incorporate prior knowledge about NPs. One useful observation was that translational noise was language-dependent: it was very useful for Japanese, but may not be so useful for European language pairs. As with the sentence-level noise, this kind of noise reduction algorithm suggests that they should be selectively applied only when the level of noise is high. At the same time, the required resources and overall computational complexity for extracting translational noise becomes considerably higher.

In Chapter 4 on statistical smoothing methods, we explicitly assumed that the distribution which we aim at learning forms a power-law distribution. Based on this assumption, we apply the hierarchical Pitman-Yor process-based smoothing method on both the language model and the translation model. In our view, smoothing techniques resolve the underfitting state in terms of the number of samples into an equilibrium with model complexities, where such an underfitting state is an inevitable consequence of relative frequency estimates. With respect to the language model, (Teh, 2006) reported that the hierarchical Pitman-Yor process method yields comparable perfor-

mance to that of the modified Kneser-Ney method. To the best of our knowledge, there have been no reports on MT quality prior to our work. Our result shows that this indeed improves the results.

There are several avenues for further study. Firstly, although this thesis mainly studied the IBM Model 1 and the HMM model, we did not focus much on IBM Model 4. What is important in IBM Model 4 is the mechanism of fertility and null insertion which extends the *pair assumption*, enabling comparison of two sentences with different lengths. This work will relate to Gibbs sampling (Geman and Geman, 1984) and / or graphical models (I.Jordan(Ed.), 1999; Bishop, 2006; Koller and Friedman, 2009). Under the MAP formulation, Gibbs sampling will be the major approach to train the model. However, it is also known that it is often difficult to obtain stable results using Gibbs sampling (Geman and Geman, 1984).

Secondly, in Section 3.1 we discussed the fact that word / phrase pairs are only extracted from the within-sentence pairs by most word alignment algorithms. This is a problem in that if just two words are observed in separate sentences, their probability will not be considered, i.e. the index of the matrix does not include them from the beginning, or just award a value of zero. Thinking about the performance on the test set, this may be a restriction. One approach to remedy this would be to employ semantic knowledge derived from language resources such as WordNet (Miller, 1995) and others.

Thirdly, as is mentioned in Bayesian HMM, many approaches in the Machine Learning community have tried to solve the problem of overfitting. Since we cannot use one prior for more than one purpose at the same time, if we use the prior for overfitting, we cannot use the prior for embedding prior knowledge. It is worthwhile to consider both of these at the same time.

Bibliograpy

References

Hirotugu Akaike. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

J. K. Baker. 1975. The dragon system – an overview. *IEEE Transactions on Acoustic Speech Signal Processing*, ASSP-23(1):24–29.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with biingual parallel corpora. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 597–604.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathematical Statistics*, 41(1):164–171.

Matthew J. Beal. 2003. Variational algorithms for approximate bayesian inference. *PhD Thesis at Gatsby Computational Neuroscience Unit, University College London*.

Christopher M. Bishop. 2006. Pattern recognition and machine learning. *Springer*.

Phil Blunsom and Trevor Cohn. 2006. Discriminative word alignment with Conditional Random Fields. *In Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL-06)*, pages 65–72.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. 1992. A training algorithm for optimal margin classifiers. *In the Proceedings of the 5th Annual ACM Workshop on COLT*, pages 144–152.

Jim Breen. 1999. A WWW Japanese dictionary. *Japanese Studies Centre Symposium*.

Peter F. Brown, Vincent J.D Pietra, Stephen A.D.Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics, Vol.19, Issue 2*, pages 263–311.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2008)*.

Nello Cristianini and John Shawe-Taylor. 2000. Introduction to Support Vector Machines. *Cambridge University Press*.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, pages 1–38.

C. Dyer, S. Muresan, and P. Resnik. 2008. Generalizing word lattice translation. *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 1012–1020.

David Forsyth and Jean Ponce. 2003. Computer vision. *Pearson*.

George Foster, Roland Kuhn, and Howard Johnson. 2006. Phrasetable smoothing for statistical machine translation. *In Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 53–61.

Alexander Fraser and Daniel Marcu. 2007. Measuring word alignment quality for statistical machine translation. *Computational Linguistics, Squibs and Discussion*, 33(3):293–303.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, Takehito Utsuro, Terumasa Ehara, Hiroshi Echizen-ya, and Sayori Shimohata. 2010. Overview of the patent translation task at the NTCIR-8 workshop. *In Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-lingual Information Access*, pages 293–302.

William Gale and Ken Church. 1991. A program for aligning sentences in bilingual corpora. *Association for Computational Linguistics*, pages 177–184.

S. Geman and D. Geman. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Zoubin Ghahramani. 2001. An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):9–42.

Sharon Goldwater, Tom L. Griffiths, and Marc Johnson. 2006. Contextual dependencies in unsupervised word segmentation. *In Proceedings of Conference on Computational Linguistics / Association for Computational Linguistics (COLING-ACL06)*, pages 673–680.

D.M. Green and J.M. Swets. 1966. Signal detection theory and psychophysics. *New York: John Wiley and Sons Inc.*

Songang Huang and Steve Renals. 2009. Hierarchical bayesian language models for conversational speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 18(8):1941–1954.

Michael I.Jordan(Ed.). 1999. Learning in graphical models. *Cambridge MA: MIT Press*.

H. Johnson, J. Martin, G. Foster, and R. Kuhn. 2007. Improving translation quality by discarding most of the phrasetable. *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 967–975.

Daniel Jurafsky and James H. Martin. 2009. Speech and language processing: An introduction to natural language processing, speech recognition, and computational linguistics. 2nd edition. *Prentice-Hall*.

Matti Kaariainen. 2009. Sinuhe – statistical machine translation using a globally trained conditional exponential family translation model. *In Proceedings of the Empirical Methods in Natural Language Processing (EMNLP09)*, pages 1027–1036.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for n-gram language modeling. *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 181–184.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. *In Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Philipp Koehn, Franz Och, and Daniel Marcu. 2003. Statistical phrase-based translation. *In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computationa Linguistics (HLT / NAACL 2003)*, pages 115–124.

P. Koehn, A. Axelrod, A. B.Mayne, C. Callison-Burch, M. Osborne, D. Talbot, and M. White. 2005. Edinburgh system description for the 2005 nist mt evaluation. *In Proceedings of Machine Translation Evaluation Workshop*.

Philipp Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for Statistical Machine Translation. *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. *In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 388–395.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. *In Proceedings of the Machine Translation Summit*, pages 79–86.

Philipp Koehn. 2010. Statistical machine translation. *Cambridge University Press*.

Daphne Koller and Nir Friedman. 2009. Probabilistic graphical models: Principles and techniques. *MIT Press*.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. *In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 24–31.

Julian. Kupiec. 1993. An algorithm for finding Noun phrase correspondences in bilingual corpora. *In Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 17–22.

Patrik Lambert, Adria de Gispert, Rafael Banchs, and Jose B. Marino. 2006. Guidelines for word alignment evaluation and manual alignment. *Language Resources and Evaluation (Springer)*, 39(4):267–285.

Z. Li, C. Callison-Burch, C. Dyer, S. Khudanpur, L. Schwartz, W. Thornton, J. Weese, and O. Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. *In Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139.

Yanjun Ma, Tsuyoshi Okita, Ozlem Cetinoglu, Jinhua Du, and Andy Way. 2009. Low-resource Machine Translation using MaTrEx: the DCU Machine Translation system for IWSLT 2009. *In Proceedings of the International Workshop on Spoken Language Translation (IWSLT 2009)*, pages 29–36.

Christopher D. Manning and Hinrich Schutze. 1999. Foundations of statistical natural language processing. *Cambridge, MA: MIT Press*.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. *In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 133–139.

Geoffrey J. McLachlan and Thriyambakam Krishnan. 1997. The EM algorithm and extensions. *Wiley Series in probability and statistics*.

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. 1953. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Daichi Mochihashi and Eiichiro Sumita. 2007. The infinite Markov model. *In Proceedings of the 20th Neural Information Processing Systems (NIPS 2007)*, pages 1017–1024.

Daichi Mochihashi, T. Yamada, and N. Ueda. 2009. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. *In Proceedings of Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2009)*, pages 100–108.

Robert C. Moore, Wen tau Yih, and Andreas Bode. 2006. Improved discriminative bilingual word alignment. *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics*, pages 513–520.

Kevin P. Murphy. 2007. Lecture slides at university of british columbia. *http://www.cs.ubc.ca/ murphyk/*.

Hermann Ney, Sonja Niesen, Franz Josef Och, Hassan Sawaf, Christoph Tillmann, and Stepan Vogel. 2000. Algorithms for statistical translation of spoken language. *IEEE Transactions on Speech and Audio Processing*, 8(1).

Franz Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. *In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Tsuyoshi Okita and Andy Way. 2010. Pitman-Yor process-based language model for Machine Translation. *International Journal on Asian Language Processing*, 21(2):57–70.

Tsuyoshi Okita, Yvette Graham, and Andy Way. 2010a. Gap between theory and practice: Noise sensitive word alignment in machine translation. *In Proceedings of the Workshop on Applications of Pattern Analysis (WAPA2010). Cumberland Lodge, England.*

Tsuyoshi Okita, Jie Jiang, Rejwanul Haque, Hala Al-Maghout, Jinhua Du, Sudip Kumar Naskar, and Andy Way. 2010b. MaTrEx: the DCU MT System for NTCIR-8. *In Proceedings of the MII Test Collection for IR Systems-8 Meeting (NTCIR-8), Tokyo.*, pages 377–383.

Tsuyoshi Okita. 2009a. Data cleaning for word alignment. *In Proceedings of Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2009) Student Research Workshop*, pages 72–80.

Tsuyoshi Okita. 2009b. Preprocessing method for word alignment. *CLUKI colloquim, DCU, Dublin*.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2002. BLEU: A Method For Automatic Evaluation of Machine Translation. *In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*.

Jim Pitman. 1995. Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 102:145–158.

Lawrence R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14:465–471.

Gideon E. Schwarz. 1978. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464.

Andreas Stolcke. 1994. Bayesian learning of probabilistic language models. *Doctoral dissertation (Department of Electrical Engineering and Computer Science University of California at Berkeley)*.

Andreas Stolcke. 2002. SRILM – An extensible language modeling toolkit. *In Proceedings of the International Conference on Spoken Language Processing*, pages 901–904.

Yee Whye Teh. 2006. A hierarchical bayesian language model based on pitman-yor processes. *In Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL-06), Prague, Czech Republic*, pages 985–992.

Vladimir Vapnik. 1998. Statistical learning theory. *Wiley and Sons*.

Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-Based word alignment in statistical translation. *In Proceedings of the 16th International Conference on Computational Linguistics (COLING 1996)*, pages 836–841.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. *In Proceedings of Human Language Technology Conference / North American Chapter of the ACL*.

Appendix A

Pseudocodes

Listing A.1: wordAlignment.m

```
while (iteration < ITERATION_MAX),
  n = size(a1F,1);
  m = size(a2F,1);
  clear total_cache;
  clear count_cache;
  clear count;
  clear total;

  count=zeros(n,m);
  total=zeros(1,m);
  logLikelihood = [];
  for i=1:size(a1,1),
    m1=size(unique(a1{i}),1);
    m2=size(unique(a2{i}),1);
    ind1 = [a1{i}];
    ind2 = [a2{i}];
    t_cache = t(unique(ind1),unique(ind2));
    total_cache=total(unique(ind2));
    count_cache=count(unique(ind1),unique(ind2));
    total_s = sum(t_cache,2);
    histgram0 = histc(a1{i},unique(a1{i}));
    histgram1 = histc(a2{i},unique(a2{i}));
    count0=ones(m1,m2);
    for j=1:max(max(histgram0),max(histgram1)),
        ind3=(histgram0==j);
        xxx3 = repmat(ind3,1,m2);
```

```
        ind4=(histgram1==j);
        xxx4 = repmat(ind4',m1,1);
        xxx5 = xxx3 | xxx4;
        count0(xxx5)=j;
    end

    count_cache = count_cache + (count0 .* t_cache) ./ repmat(total_s,
1,m2);
    total_cache = total_cache + sum((count0 .* t_cache) ./ repmat(
total_s,1,m2), 1);
    count(unique(ind1),unique(ind 2)) = count_cache;
    total(unique(ind2))=total_cache;
  end

  t = zeros(size(a1F,1),size(a2F,1));
  t = count ./ repmat(total,n,1);
  ind = (t < PROB_SMOOTH) & (t ~= 0);
  t(ind)=PROB_SMOOTH;
  ind = t > 0;
  logLikelihood = [logLikelihood sum(log2(t(ind)))];
  iteration = iteration + 1;
end
```

Listing A.2: trainingVBEM.py

```
def trainVBEM(L1tok,L2tok):
    ...the same as trainStandardEM ...
    while (iteration < ITERATION_MAX):
        count = {}
        total = {}
        for i in range(numSentence):
            ... set the prior ...
            for e in L1tok[i]:
                total_s[e] = 0
                for f in L2tok[i]:
                    try:
                        total_s[e] += t[(e,f)]
                    except:
                        pass
            for e in L1tok[i]:
                for f in L2tok[i]:
                    try:
                        count[(e,f)] += t[(e,f)] / total_s[e]
                    except KeyError:
                        try:
```

```
                            count[(e,f)] = t[(e,f)] / total_s[e]
                        except:
                            pass
                    try:
                        total[f] += t[(e,f)] / total_s[e]
                    except KeyError:
                        try:
                            total[f] = t[(e,f)] / total_s[e]
                        except:
                            pass
        ... calculation of the lower bound ...
        for x in count.keys():
            f=x[1]
            try:
                pri=prior[f]
            except:
                pri=1.0
            try:
                t[x] = count[x] / total[f] * pri
            except KeyError:
                pass
        logLikelihood = -sum(log(x[1]) for x in t.items())

        for x in count.keys():
            f=x[1]
            if (t[x] < PROB_SMOOTH):
                t[x]=PROB_SMOOTH
    ...the same as trainStandardEM ...
```

Listing A.3: Baum-Welch Algorithm

```
def baumWelch(hmm, obs_seqs, **args):
    epochs = 20
    scaling=1
    normUpdate=1
    verbose=1
    K = len(obs_seqs)
    start = time.time()
    LLs = []
    for epoch in xrange(epochs):
        start_epoch = time.time()
        LL_epoch =0
        E_si_all = zeros([hmm.N], float)
        E_si_all_TM1 = zeros([hmm.N], float)
        E_si_sj_all = zeros([hmm.N,hmm.N], float)
```

```
        E_si_sj_all_TM1 = zeros([hmm.N,hmm.N], float)
        E_si_t0_all = zeros([hmm.N])
        eEmitProb = zeros([hmm.N,hmm.M], float)
        ow = 0
        for obs in obs_seqs:
            obs = list(obs)
            logProbObs, alpha, c = forward(hmm=hmm, obs=obs, scaling=1)
            beta = backward(hmm=hmm, obs=obs, c=c)
            LL_epoch += logProbObs
            T = len(obs)
            if normUpdate:
                print logProbObs, log(len(obs))
                w_k = 1.0 / -(logProbObs + log(len(obs)))
            else:
                w_k = 1.0
            obs_symbols = obs[:]
            obs = symbol_index(hmm, obs)
# gamma[i,t] = P(q_t = S_i|obs,hmm)
            gamma_raw = alpha * beta
            gamma = gamma_raw / gamma_raw.sum(0)
            E_si_t0_all += w_k * gamma[:,0]
            E_si_all += w_k * gamma.sum(1)
            E_si_all_TM1 += w_k * gamma[:,:T-1].sum(1)
# xi[i,j,t] = P(q_t = S_i, q_t+1 = S_j|obs, hmm)
            xi = zeros([hmm.N,hmm.N, T-1], float)
            for t in xrange(T-1):
                for i in xrange(hmm.N):
                    xi[i,:,t] = alpha[i,t] *hmm.transProb[i,:] *
hmm.emitProb[:, obs[t+1]] * beta[:,t+1]
                if not scaling:
                    xi[:,:,t] = xi[:,:,t] / xi[:,:,t].sum()
            E_si_sj_all += w_k * xi.sum(2)
            E_si_sj_all_TM1 += w_k * xi[:,:,:T-1].sum(2)
            emitProbNew = zeros([hmm.N,hmm.M], float)
            for k in xrange(hmm.M):
                which = array([hmm.symbol[k] == x for x in obs_symbols])
                emitProbNew[:,k] = gamma.T[which,:].sum(0)
            eEmitProb += w_k * emitProbNew
        E_si_t0_all = E_si_t0_all / sum(E_si_t0_all)
        hmm.Pi = E_si_t0_all
        transProbNew = zeros([hmm.N,hmm.N], float)
        for i in xrange(hmm.N):
            transProbNew[i,:] = E_si_sj_all_TM1[i,:] / E_si_all_TM1[i]
        hmm.transProb = transProbNew
        for i in xrange(hmm.N):
```

```
            eEmitProb[i,:] = eEmitProb [i,:] / E_si_all[i]
        hmm.emitProb = eEmitProb
        LLs.append(LL_epoch)
    return hmm, LLs
```

Listing A.4: fwdBack.py

```
def forward(hmm, obs, scaling=True):
    T = len(obs)
    print 'obs',obs
    obs = symbol_index(hmm, obs)
    print 'sym',obs
    if scaling:
        c = zeros([T], float)
    alpha = zeros([hmm.N,T], float)
    alpha[:,0] = hmm.Pi * hmm.emitProb[:,obs[0]]
    if scaling:
        c[0] = 1.0 / sum(alpha[:,0])
        alpha[:,0] = c[0] * alpha[:,0]
    for t in xrange(1,T):
        alpha[:,t] = dot(alpha[:,t-1],hmm.transProb)*hmm.emitProb

[:,obs[t]]
        if scaling:
            c[t] =  1.0 / sum(alpha[:,t])
            alpha[:,t] = alpha[:,t]  * c[t]
    print 'alpha=',alpha
    if scaling:
        logProbObs = -(sum(log(c)))
        return (logProbObs, alpha, c)
    else:
        probObs = sum(alpha[:,T-1])
        return (probObs, alpha)

def backward(hmm, obs, c=None):
    T = len(obs)
    obs = symbol_index(hmm, obs)
    beta = zeros([hmm.N, T], float)
    beta[:, T-1] = 1.0
    if c is not None:
        beta [:,T-1 ] = beta [:,T-1] * c[T-1]
    for t in reversed(xrange(T-1)):
        beta[:,t] = dot(hmm.transProb,(hmm.emitProb[:,obs[t+1]] *
beta[:,t+1]))
        if c is not None:
```

```
            beta[:,t] = beta[:,t] * c[t]
    return beta
```

Listing A.5: viterbi.py

```
def viterbi(hmm, obs, scaling=True):
    T = len(obs)
    obs = symbol_index(hmm, obs)
    delta = zeros([hmm.N,T], float)
    if scaling:
        delta[:,0] = log(hmm.Pi) + log(hmm.emitProb[:,obs[0]])
    else:
        delta[:,0] = hmm.Pi * hmm.emitProb[:,obs[0]]
    psi =  zeros([hmm.N, T], int)
    if scaling:
        for t in xrange(1,T):
            nus =  delta[:,t-1] + log(hmm.transProb)
            delta[:,t] = nus.max(0) + log(hmm.emitProb[:,obs[t]])
            psi[:,t] = nus.argmax(0)
    else:
        for t in xrange(1,T):
            nus =  delta[:,t-1] * hmm.transProb
            delta[:,t] = nus.max(0) * hmm.emitProb[:,obs[t]]
            psi[:,t] = nus.argmax(0)
    qStar =  [argmax(delta[:,T-1])]
    for t in reversed(xrange(1, T-1)) :
        qStar.insert(0, psi[qStar[0],t+1])
    return (qStar, delta, psi)
```

Table 1 shows the prior knowledge about alignment links used in the paraphrase examples in Section 1 (Refer to Figure 1.3.)

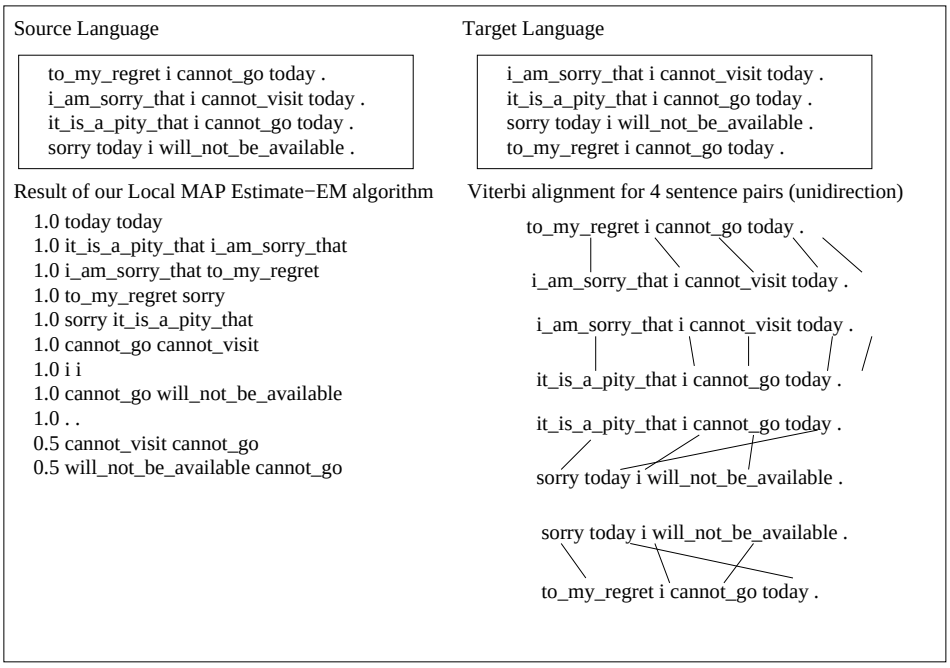

Figure A.1: The result of word alignment by our local MAP estimate-EM aligner removing two commas from parallel corpus. (Compare this result with the result shown in Figure 1.3. The ambiguous phrase pairs whose probability are all 0.0001 were not yielded in this case.)