UNIVERSITY OF SOUTHAMPTON

Large-scale Reordering Models for Statistical Machine Translation

by

Abdullah Saleh A. Alrajeh

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Engineering, Science and Mathematics

School of Electronics and Computer Science

May 2015

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Abdullah Saleh A. Alrajeh

In state-of-the-art phrase-based statistical machine translation systems (SMT), modelling phrase reorderings is an important need to enhance naturalness of the translated outputs, particularly when the grammatical structures of the language pairs differ significantly. The challenge in developing machine learning methods for machine translation can be summarised in two points. First is the ability to characterise language features such as morphology, syntax and semantics. Second is adapting complex learning algorithms to process large corpora.

Posing phrase movements as a classification problem, we exploit recent developments in solving large-scale SVM, Multiclass SVM and Multinomial Logistic Regression. Using dual coordinate descent methods for learning, we provide a mechanism to shrink the amount of training data required for each iteration. Hence, we produce significant saving in time and memory while preserving the accuracy of the models. These efficient classifiers allow us to build large-scale discriminative reordering models. We also explore a generative learning approach namely naive Bayes. Our Bayesian model is shown to be superior to the widely-used lexicalised reordering model. It is fast to train and the storage requirement is many times smaller than the lexicalised model. Although discriminative models might achieve higher accuracy than naive Bayes, the absence of iterative learning is a critical advantage for very large corpora.

Our reordering models are fully integrated with the Moses machine translation system, widely used in the community. Evaluated in large-scale translation tasks, our models have proved successful for two very different language pairs: Arabic-English and German-English.

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Nomenclature

Symbol	Notation
$\mathbf{f}/\mathbf{e}$	a source / target sentence (string)
$\mathbf{f}^i/\mathbf{e}^i$	the i-th source / target sentences in the data set
f_j	the j-th word in the source sentence
e_i	the i-th word in the target sentence
$\bar{\mathbf{f}}^I/\bar{\mathbf{e}}^I$	a source / target phrase sequence
$\bar{f}_j$	the source phrase where $\bar{f}$ denotes the sequence of words and j denotes $\bar{f}$ is the j-th phrase in the source phrase sequence $\bar{\mathbf{f}}^I$
$\bar{e}_i$	the source phrase where $\bar{e}$ denotes the sequence of words and i denotes $\bar{e}$ is the i-th phrase in the source phrase sequence $\bar{\mathbf{e}}^I$
$\boldsymbol{S}$	a set of examples
$\mathbf{x}_i$	the input of the i-th example in $\boldsymbol{S}$
$\mathbf{y}_i$	the output of the i-th example in $\boldsymbol{S}$
$\mathbf{Y}$	the output space $\mathbf{y}_i \in \mathbf{Y}$
$\mathbb{F}$	the input (source string) space
$\mathbb{E}$	the output (target string) space
$\Delta(\mathbf{y}_i, \mathbf{y})$	the "distance" between the true output and the pseudo candidate
N	the number of examples in $\boldsymbol{S}$
$\phi(\mathbf{x})$	the feature vector of input $\mathbf{x}$
$\phi(\mathbf{x}, \mathbf{y})$	a joint feature vector
d	the dimension of feature space
d	the phrase reordering distance
o	the phrase reordering orientation
$\boldsymbol{O}$	the set of phrase reordering orientations $o \in \boldsymbol{O}$
dim	the dimension of
$\mathbf{w}$	a weight vector $\mathbf{w} \in \mathbb{R}^d$
$\mathbf{w}_o$	a weight vector measuring features' contribution to orientation o
$\{\mathbf{w}_o\}_{o \in \boldsymbol{O}}$	the set of weight vectors for the phrase reordering model
$f(\mathbf{x})$	a linear evaluation function $f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$
$\boldsymbol{S}_{\bar{f}}$	a set of phrase pairs $(\bar{f}_j/\bar{e}_i)$ shared the same source phrase $\bar{f}$
$(\bar{f}_j^n, \bar{e}_i^n)$	the n-th example in $\boldsymbol{S}_{\bar{f}}$ that is also abbreviated as $(\bar{f}^n, \bar{e}^n)$
$\phi(\bar{f}^n, \bar{e}^n)$	the feature vector of the phrase pair $(\bar{f}^n, \bar{e}^n)$

a	an alignment function maps each f_j to e_i in a sentence pair
$a(i\|j, l_f, l_e)$	a probability distribution of a position of e_i given a position of f_j and the lengths of the sentence pair
$\varnothing_i$	the fertility of e_i which is how many f words could be aligned to e_i
$n(\varnothing_i\|e_i)$	a probability distribution of the fertility

Declaration of Authorship

I, Abdullah Alrajeh , declare that the thesis entitled *Large-scale Reordering Models for Statistical Machine Translation* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: (Alrajeh and Niranjan, 2014a), (Alrajeh and Niranjan, 2014b), (Alrajeh et al., 2015), (Alrajeh and Niranjan, 2015a) and (Alrajeh and Niranjan, 2015b).

Signed: . . . . . .

Date: . . . . . .

Acknowledgements

I would like to thank my main supervisor Mahesan Niranjan for his invaluable support and guidance throughout this journey. I also would like to thank my second supervisor Srinandan Dasmahapatra for the useful comments and suggestions.

Thanks to my colleagues and friends at the University of Southampton, particularly the Vision, Learning and Control (VLC) research group. Many thanks to Yizhao Ni for the helpful discussions at an early stage of this research.

I am very grateful to my parents and my wife, whom I owe a lot.

Finally, I acknowledge and thank King Abdulaziz City for Science and Technology (KACST) for funding my scholarship.

To my son, Yusuf.

Chapter 1

Introduction

Machine translation is one of the oldest and hardest problems in artificial intelligence. Efforts to solve this problem started from the early days of computers. One of the main motivations for machine translation is the interest of intelligence agencies to multiply their ability to know what is happening overseas. Although machine translation has a long history, full automatic translation of high quality seems hard to achieve at least in the near future. The translation problem is related to many language and cultural issues that makes it a very hard problem.

Translation is a process of transferring the meaning of words or text to another language. It involves decoding the meaning of the source language and then re-encoding that meaning into the target language. The process generally involves a set of complex operations. It requires a full knowledge of the source language which includes: morphology, syntax, semantics and pragmatics. The context that surrounds the translated text should be considered as an isolated sentence might have different meanings. It also requires the same in-depth knowledge for the target language to re-encode the meaning.

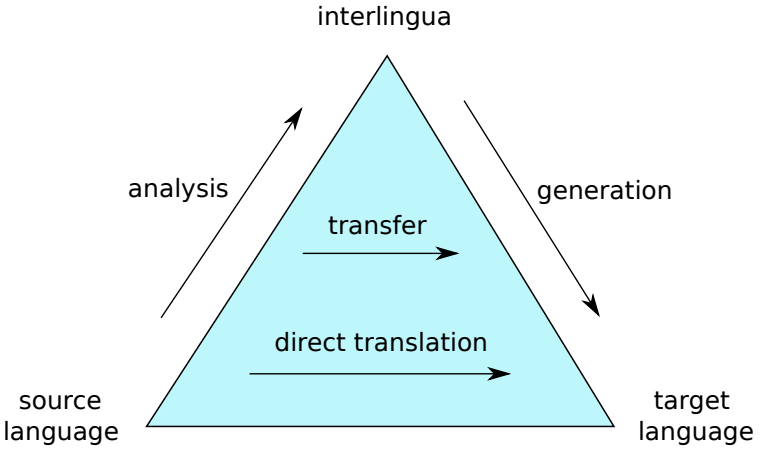

FIGURE 1.1: Translation process in different methods.

Figure 1.1 is a standard diagram shows the translation process from a source language into a target language in three different methods. First is direct translation which does not require any text analysis because words are translated literally with basic rules. Second is transfer method which uses morphological and syntactic analysis. Third is the most difficult method. It is called Interlingua which is an international language represents the abstract meaning of the text.

1.1 History of Machine Translation

From the early days until now, machine translation was generally funded by governments for national security issues. In 1954, Georgetown University and IBM developed a system that can translate from Russian to English using limited vocabulary and six grammar rules (Slocum, 1985) in order to attract government agencies. The experiment succeeded in fascinating both the government and the public for the prospect of machine translation. The developers claimed that within a few years, the problem of machine translation could be solved. However, progress was very slow. After more than a decade, US funding agencies commissioned the Automatic Language Processing Advisory Committee (ALPAC) for investigation. ALPAC report in 1966 showed that the capabilities of machine translation had been over-promised and full human translation was cheaper than post-editing the output of machine translation. As consequence, funding for machine translation was mostly stopped.

1.1.1 Meaning-Oriented Machine Translation

In the 1980s and 1990s, researchers focused on systems that use intermediate representations of the meaning. Interlingua is an extreme case of meaning-oriented systems. It uses an international language to represent the abstract meanings independent of a specific language. The problem is attractive but is considered a grand challenge in artificial intelligence.

A rule-based approach is less complicated than Interlingua. It has intermediate representations for morphology, syntax and semantics. They are collected and revised manually by linguistics experts. Rule-based systems are effective because much of the linguistics knowledge is static. However, they suffer from two major limitations. First, construction of linguistics rules is expensive and time consuming. Second, adding new rules may conflict with other rules. Solving these problems takes a long time which affects the efficiency. Many commercial systems were developed using rule-based approaches such as Systran and Logos.

1.1.2 Data-Driven Machine Translation

In the late 1980s and early 1990s, there has been more interest in data-driven machine translation which was a major turning point. It is sometimes called corpus-based, analogy-based, memory-based or experience-guided translation.

Example-based is one of the early suggested approaches in data-driven machine translation. It tries to find a similar sentence for a given input in previously translated examples. Then it makes proper changes for the selected translation. This approach is faster than rule-based but does not guarantee better translations.

Due to the increase in computing power and the accessibility of huge data in the public domain, a statistical approach was suggested to make deeper analysis than the example-based approach. Brown et al. (1988) from IBM present the mathematics of statistical machine translation. Later, Brown et al. (1993) introduced five statistical models known as IBM models and gave algorithms for estimating their parameters. Although they put machine translation on a solid mathematical foundation, it seems that the world was not ready for it. Most researchers were focusing on other approaches.

After a decade, the statistical approach gathered a strong momentum. In 1999, many researchers came together at a summer workshop in Johns Hopkins University to re-implement most of the IBM methods. Al-Onaizan et al. (1999) in that six-week summer workshop implemented statistical machine translation (SMT) toolkit (called EGYPT) and one of the tools mentioned in their technical report was GIZA for word alignment. Franz Och, who was one of the participants, later extended the word alignment tool to GIZA++ that added a lot of additional features. This tool becomes one of the main blocks in several SMT systems including ours.

1.1.3 Current Developments

Many academic and commercial research labs are currently developing statistical machine translation (SMT) systems. Large companies such as IBM, Google and Microsoft offer translation services using SMT systems. In fact, traditional machine translation companies such as Systran have started to integrate statistical methods to their systems.

Recently, machine translation is again taking a new direction towards deeper analysis of the data. Statistical methods have humble capabilities for some aspects of natural languages such as the word order problem (i.e. source and target words are not in synchrony) when translating between two grammatically different languages (e.g. Arabic-English). Machine learning technologies are able to explore the complexity of these problems and formulate a mathematical mapping between these languages.

A lot of workshops are held every year to discuss the latest developments in machine translation. Some of these workshops are primarily for evaluating the output of SMT systems. For example, NIST Open Machine Translation (OpenMT) is an evaluation workshop started in 2001 and the latest one was in 2012. The next OpenMT workshop will take place in the spring of 2015. Participants in these events report the most useful techniques in building their translation systems. These reports are a great resource for interesting research ideas.

1.2 Thesis Objectives

Currently, the dominant approach to machine translation is statistical. The mathematical basis of this approach has its origins in the formulation due to Brown et al. (1988), who later introduced five statistical models widely known as the IBM models (Brown et al., 1993). While these early models were word-based, assuming the translation to take place on a word by word basis, in reality, groups of words (phrases) are recognised as better units of translation (Koehn, 2010a).

While such attempts at phrase level translation has shown improvement in translation performance, a further issue that has to be addressed is that of long range phrase reorderings (Galley and Manning, 2008). Such reorderings arise from differences in grammatical structures between language pairs and addressing this is important in achieving increased naturalness of the translated output (Koehn, 2010a). This issue is particularly pronounced when language pairs separated by large evolutionary distances, or from different linguistic families, are considered such as Arabic and English.

Analogous to speech recognition systems, translation systems relied on language models to produce more fluent translation. While early work on handling phrase reorderings implemented a relaxation into the decoder which, instead of forcing phrases to be in synchrony, allowed a penalty function that penalised large movements proportionately (Koehn, 2004a). An alternative approach, adopted by several systems nowadays is lexicalised reordering modelling (Tillmann, 2004; Kumar and Byrne, 2005; Koehn et al., 2005), whereby the frequencies of relative positions of the phrase pairs are extracted from the training corpus and used as additional inputs to the decoder. These approaches may suffer from the data sparseness problem since many phrase pairs occur only once (Nguyen et al., 2009).

Building on this, some researchers have borrowed powerful ideas from the machine learning literature, to pose the phrase movement problem as a prediction problem using contextual input features whose importance is modelled as weights of a linear classifier trained by maximum entropy (MaxEnt) method which is a popular choice (Zens and Ney, 2006; Xiong et al., 2006; Nguyen et al., 2009; Xiang et al., 2011). Other methods have also been used to model long distance phrase movements. Ni et al. (2011) propose

a learning approach based on a variant of a perceptron which differs from previous works in that training data were divided into small independent sets where all samples share the same source phrase were considered a training set. This method breaks down the learning complexity to have as many sub-models as source phrases.

While a large-scale parallel corpus is advantageous for improving such reordering model, this improvement comes at the price of computational complexity. This issue is particularly pronounced when discriminative models are considered such as maximum entropy-based model due to many iterations over the dataset required for learning.

The aim of the work in this thesis is to explore recent advancements in solving large-scale classification problems in order to produce significant saving in computation and memory while preserving the accuracy of the translation task. We also aim to combine the advantages of lexicalised and discriminative reordering models using a generative modelling approach.

1.3 Contributions of the Thesis

There are six main contributions in this thesis. A summary of these contributions is provided here, with a more thorough description given in Chapter 8.

- We propose computationally fast and memory-efficient algorithms to learn SVM, Multiclass SVM and Multinomial Logistic Regression. Using dual coordinate descent methods for learning, we provide a mechanism to shrink the amount of training data required for each iteration. Hence, we produce significant computational saving while preserving the accuracy of the models. These efficient classifiers allow us to build large-scale discriminative reordering models. Experiments were carried out on a parallel corpus with more than a quarter of a billion words.

- We empirically show that the approach of Ni et al. (2011) to break down the learning complexity into small sub-models is not necessary. In fact, having one reordering model is more beneficial to a machine translation system. Although the number of parameters for each sub-model is small, the overall parameters are larger than having one model that incorporates all the training data.

- We explore a generative learning approach to phrase reordering namely naive Bayes. Our Bayesian model using a Dirichlet prior is shown to be superior to the lexicalised model of estimating probabilities as relative frequencies of phrase movements. The training time of naive Bayes is as fast as the lexicalised model and its storage requirement is many times smaller. Discriminative models might achieve higher score than naive Bayes. However, its parameter estimation requires only one pass over the data with limited memory (i.e. no iterative learning). This

is a critical advantage over discriminative models particularly for very large corpora. To the best of our knowledge, this model of classification has not been used in this context previously.

- Our reordering models are fully integrated with the widely used open source system Moses (Koehn et al., 2007). Unlike Distance Phrase Reordering (DPR) package (Ni et al., 2010a), our code takes its input (i.e. phrases and their movements) directly from Moses's tools. This important change enables it to benefit from new updates to Moses regarding phrase extraction. In addition to that, DPR package is not practical for online translation tasks. It computes reordering probabilities outside Moses's decoder which means each time there is a new text the package has to be run again.

- Ni (2010) suggests to extend the formulation of phrase reordering from classification problem to ordinal regression. The argument is that phrases exist in order and such a model should respect this structure. A classification model with flexible margins between classes is a step towards this direction (Ni et al., 2011). In our work, we found ordinal regression empirically not useful. It has low accuracy compared to SVM and Multinomial Logistic Regression.

- Farran and Saunders (2009) introduce an online algorithm called Voted Spheres. Its non-parametric nature is attractive. It was proposed as a large-scale classifier. The experiments showed that the algorithm cannot scale to our problem (i.e. millions of phrase pairs). We found that the classifier has high accuracy. However, it is very slow during prediction phase (i.e. 100 times or more slower than competing classifiers) which makes it not practical for machine translation systems.

1.4 Publications and Software

The work in this thesis has led to a several peer-reviewed publications with one journal paper:

- A. Alrajeh, and M. Niranjan (2014) Large-scale reordering model for statistical machine translation using dual multinomial logistic regression. *In, Empirical Methods on Natural Language Processing, Qatar.*

- A. Alrajeh, and M. Niranjan (2014) Bayesian reordering model with feature selection. *In, ACL: The Ninth Workshop on Statistical Machine Translation, USA.*

- A. Alrajeh, A. Takeda, and M. Niranjan (2014) Memory-efficient large-scale linear support vector machine. *In, The 7th International Conference on Machine Vision, Italy.*

- A. Alrajeh, and M. Niranjan (2015) Generative and discriminative reordering models for statistical machine translation. *In, The 8th Saudi Students Conference, United Kingdom.*

- A. Alrajeh, and M. Niranjan (2015) Scalable reordering models for SMT based on multiclass SVM. *The Prague Bulletin of Mathematical Linguistics.*

Software

Two tools [1] were developed in order to support the experimental studies in this thesis:

- *Large-scale Classification Tool (LCT)*:
  LCT is a C++ tool implementing several classifiers (e.g. SVM, MaxEnt) able to handle millions of training samples efficiently.

- *Scalable Reordering Models (SRM)*:
  SRM is a fully integrated package with Moses translation system implements advanced machine learning techniques to learn phrase reordering models. The package is capable of learning these models from millions of parallel sentences.

1.5 Thesis Outline

The rest of the thesis is structured as follows:

- Chapter 2 gives a background about statistical machine translation and introduces the fundamental concepts.

- Chapter 3 describes the phrase reordering problem in machine translation and extensively reviews the previous work to tackle it.

- Chapter 4 presents in the first part different ways of extracting features from a parallel corpus for modelling phrase movements. A novel phrase representation is also presented which we found to be effective. The second part discusses several methods to reduce feature space.

- Chapter 5 gives a short introduction to machine learning. It also explores several machine learning algorithms to learn classification models.

- Chapter 6 describes our novel techniques for efficiently learning large-scale classification models. The chapter covers both generative and discriminative approaches.

[1] They will be publicly available on `https://github.com/asrajeh`

- Chapter 7 evaluates the classification models presented in the previous chapter on two phases. First, the classifiers are examined on benchmark datasets in the field of pattern recognition in general. Second, reordering models are built based on these classifiers from a parallel corpus.
- Chapter 8 examines the impact of reordering models in translation systems. Two machine translation tasks have been done, from Arabic to English and from German to English. The translation system are tested on well-known news translation benchmarks.
- Chapter 9 summarises the major findings of the thesis and future research directions are suggested.

Chapter 2

Statistical Machine Translation

Statistical machine translation (SMT) is an approach to machine translation based on statistical analysis of a collection of sentences pairs from two languages, parallel corpus. The mathematical basis of statistical machine translation has its origins in the formulation due to Brown et al. (1988), who later introduced five statistical models widely known as the IBM models (Brown et al., 1993). Although the state of the art much more advanced than their approach but the principles are still in use. They applied those models to French and English corpus from the proceedings of the Canadian Parliament but because the linguistic content in the algorithms kept to minimum, they can be applied to any pairs of languages. Since they worked on French-English corpus, $\mathbf{f}$ was denoted as a source sentence consists of J words (i.e. $\mathbf{f} = f_1, \ldots, f_J$) and $\mathbf{e}$ as a target sentence consists of I words (i.e. $\mathbf{e} = e_1, \ldots, e_I$).

Brown et al. (1993) suggest that the most likely translation of a French sentence $\mathbf{f}$ can be found by seeking the English sentence $\mathbf{e}$ that maximises the conditional probability $p(\mathbf{e}|\mathbf{f})$. It is written as follows:

$$\mathbf{e}_{\text{best}} = \underset{\mathbf{e}}{\operatorname{argmax}}\, p(\mathbf{e}|\mathbf{f}). \tag{2.1}$$

Since there are unlimited choices that can be considered as English sentences in this search problem (argmax), training this model is very difficult. Hence, Bayes rule has been used to break this complexity. Instead of modelling the probability of an English sentence being grammatically correct and translation of a given French sentence, Bayes rule breaks this process into two models: language model $p(\mathbf{e})$ and translation model $p(\mathbf{f}|\mathbf{e})$.

$$\underset{\mathbf{e}}{\operatorname{argmax}}\, p(\mathbf{e}|\mathbf{f}) = \underset{\mathbf{e}}{\operatorname{argmax}}\, \frac{p(\mathbf{e})p(\mathbf{f}|\mathbf{e})}{p(\mathbf{f})}. \tag{2.2}$$

The denominator here, $p(\mathbf{f})$, is independent of $\mathbf{e}$ because we are given a French sentence and looking for its translation. Therefore, the expression can be written without it as follows:

$$\underset{\mathbf{e}}{\operatorname{argmax}}\, p(\mathbf{e}|\mathbf{f}) = \underset{\mathbf{e}}{\operatorname{argmax}}\, p(\mathbf{e})p(\mathbf{f}|\mathbf{e}). \tag{2.3}$$

The search problem (argmax) now is restricted to $\mathbf{e}$ sentence that is grammatically correct because the ungrammatical sentence will get very low probability. This method is well-known in communications and called noisy-channel model.

The following example explains the noisy-channel model in the context of machine translation. Suppose you were expecting an English message from a friend and for some reason it was translated into French. To recover the original message $\mathbf{e}$, there are two things you are looking for. First, you are looking for $\mathbf{e}$ that is considered a good English sentence using a language model which is represented by $p(\mathbf{e})$. Second, you are looking for $\mathbf{e}$ that would be translated into $\mathbf{f}$ using translation model which is represented by $p(\mathbf{f}|\mathbf{e})$. These two things might conflict, so $\mathbf{e}$ maybe given a high probability in language model but low probability in translation model. The most likely $\mathbf{e}$ to be the original message is the one maximises the product of both of them.

This is a general view of how statistical machine translation systems work and the following sections will give more details. The first section is about the data that translation systems were based on. Then, the next two sections describe language and translation modelling. Section four explains how to search for the most likely translation and this process is called decoding. Section five discusses the problem of measuring translation quality. The last section is about improving SMT system by tuning (i.e. giving various weights for the system's models).

2.1 Corpora

It is very difficult to have a general-purpose translation system. The problem will be simplified dramatically with restricted domain. Hence, early developed systems were for limited domain such as weather reports and technical manuals. Currently, the development tends to open domain or at least large domain such as news.

Acquiring parallel-corpus for training is not an easy task. Lots of translated materials are not available in the research field. Recently, there are many parallel corpora became available, but in specific domains and a small number of languages. For example, EuroParl corpus is a collection of debate transcripts from the proceedings of the European Parliament. The Linguistic Data Consortium of the University of Pennsylvania distributes a variety of corpora (http://ldc.upenn.edu).

The web is a great resource for parallel texts. Many web sites publish their content in several languages. News organisations for instance have multilingual audience such as BBC. In the middle east, AL-ARABIYA web site publishes its content in four languages: Arabic, English, Farsi and Urdu.

Crawling the web for parallel data is a good solution but there are some issues that might make it difficult. One of these issues is document alignment. The structure of the web site helps to find the corresponding document in the other language but sometimes it is not straightforward. One of the early work in acquiring parallel corpora was by Resnik (1999) who uses a language independent system to find parallel text on the web. A dictionary was used by Fukushima et al. (2006) for detecting English-Japanese parallel texts. Li and Liu (2008) use similarity of the URL and page content to automatically find parallel documents. Although acquiring algorithms do most of the task however usually there is a manual effort (Martin et al., 2003; Koehn and Monz, 2005).

Parallel corpora can also be built by translating text from scratch for building translation system (Germann, 2001). Choosing the most relevant new sentences for the training data using methods such active learning improves SMT system more than random sentences. (Majithia et al., 2005).

A typical parallel corpus does not usually have sentence by sentence translation which means long sentences might be broken up and the short ones merged. Since work began on statistical machine translation, researchers started to tackle this problem which is known as sentence alignment. Early suggested methods were using sentence length (Brown et al., 1991; Gale and Church, 1993). The method is fast since it does not use any lexical information and therefore it is practical for large collections. Other methods proposed deeper analysis using lexical information such as Aswani and Gaizauskas (2005) who proposed a hybrid approach using regression techniques. They reported high accuracy for many-to-many sentence alignment. Enright and Kondrak (2007) claim a simple and fast method that depends on overlap of rare words such as cognates, names and numbers. The method might work with languages that share common origin (e.g. Romance languages) but with Arabic-English it is not simple.

There is another issue in collecting parallel texts that are not directly translated from the source side. Multilingual web sites like BBC sometimes adapt their news for different audiences. If thats mostly the case, the data will be called comparable corpus. Learning from such a corpus shows recently interesting results even if the resources are not translation of each other but share similar content. Munteanu and Marcu (2005) exploit comparable corpora to improve translation. Log-likelihood ratios have been proposed to extract parallel sentences (Munteanu et al., 2004) and sentence fragments (Munteanu and Marcu, 2006). Quirk et al. (2007) also suggest a generative model for fragment extraction (i.e. parallel phrases).

An example of automatically extracted parallel sentences is ISI Arabic-English corpus on Linguistic Data Consortium (LDC) with catalog number (LDC2007T08). It was extracted automatically from two monolingual corpora: Arabic Gigaword Second Edition (LDC2006T02) and English Gigaword Second Edition (LDC2005T12). The data was extracted from news articles published by Xinhua News Agency and Agency France Press and was obtained using the automatic parallel sentence identification method described in Munteanu and Marcu (2005). The parallel sentence identification approach is designed to judge sentence pairs in isolation from their contexts, and can therefore find parallel sentences within document pairs which are not parallel.

2.2 Language Modelling

Language modelling is a way to know how likely such a sentence would be uttered by a language speaker (e.g. English speaker). In statistical machine translation, it helps to limit the search problem for fluent output. For example, a good language model should give higher probability for "this is a new car" than "this is a car new". A good introduction for language modelling is given by Chen and Goodman (1998).

A typical approach for building a probabilistic language model is by collecting a large amount of text (i.e. monolingual corpus). Then the probability for a given sentence is by counting its occurrence in the corpus and dividing it by the total number of sentences. There is a serious problem in this approach. The corpus, even very large one, will not have all the possible sentences. Therefore, many good sentences might be assigned zero probability. Breaking down sentences into smaller parts enables us to collect sufficient statistics. This method seems similar to how our mind works. We easily recognise if a sentence is a proper one even if we have not seen it before. The mind breaks down the sentence into components and see if they combine in a reasonable way.

An easy way to break up the process is to predict one word e_i at a time by decomposing the probability $p(\mathbf{e})$ using the chain rule:

$$p(\mathbf{e}) = p(e_1, e_2, \ldots, e_I) = p(e_1)p(e_2|e_1)\ldots p(e_I|e_1, e_2, \ldots, e_{I-1}) \tag{2.4}$$

Computing each word probability given the previous ones is hard. Hence, the model estimation can be simplified by limiting the history to n words. This kind of model is called Markov chain and n is the order of the model. It assumes that only n number of previous words affect the next word probability which is called the Markov assumption. Although this assumption is technically wrong, it is more practical than computing the whole history.

$$p(e_i|e_1, e_2, \ldots, e_{i-1}) \simeq p(e_i|e_{i-n}, \ldots, e_{i-1}) \tag{2.5}$$

N-gram language models are usually estimated over 3 to 5 grams. For example, trigram model means two words history are considered for predicting the third word. Bigrams model requires just one word to estimate the next one while unigram model disregards the previous words. This unigram model (2.6) is easy to estimate but it is not a good language model. Two similar sentences with different word order will have the same probability.

$$p(\mathbf{e})_{\text{unigram}} = p(e_1)p(e_2)\dots p(e_n) \tag{2.6}$$

The following equation (2.7) is an example of how to estimate the sentence "this is a new car" according to bigram language model. The sentence probability is the product of the six grams. First estimates the chance that a sentence starts with word "this". The next one is how likely word "is" would follow word "this" and so on. Last gram is the probability of a sentence that ends with the word "car".

$$p(\mathbf{e})_{\text{bigram}} = p(\text{this}|\mathbf{start})p(\text{is}|\text{this})p(\text{a}|\text{is})p(\text{new}|\text{a})p(\text{car}|\text{new})p(\mathbf{end}|\text{car}) \tag{2.7}$$

The estimation of a bigram is by counting how many times two words found together in the corpus and dividing it by how many times the first word occurs. In general, we estimate n-gram as follows:

$$p(e_i|e_{i-n},\dots,e_{i-1}) = \frac{\text{count}(e_{i-n},\dots,e_{i-1},e_i)}{\sum_e \text{count}(e_{i-n},\dots,e_{i-1},e)}. \tag{2.8}$$

As mentioned earlier, any corpus will not have all the possible sentences. Therefore, a language model based on sentence frequency might assign zero probability to a fluent sentence because it did not occur in the corpus. N-gram models manage to avoid assigning zero probability to unseen sentences by breaking up the estimation process into n-gram. However, if there is one n-gram in a given sentence that was not in the training data, the model will assign the sentence zero probability since the estimation is based on the product of all n-grams. This case is likely to occur in higher model such as 5-gram model. Many good sentences will have unseen 5-grams, and assigning them zero probability is a harsh decision.

There are many smoothing methods have been proposed to solve this problem. These methods prevent language models from assigning zero probability for any sentence even very bad ones. Chen and Goodman (1998) did an empirical study of smoothing techniques for language modelling. A more recent work is by Zhai and Lafferty (2004). One of the simplest methods is to add 1 to every n-gram count. Another method is to break up an n-gram into smaller parts with different weights, as shown in this example:

$$p(e_3|e_1,e_2) = C_3\frac{\text{count}(e_1,e_2,e_3)}{\text{count}(e_1,e_2)} + C_2\frac{\text{count}(e_2,e_3)}{\text{count}(e_2)} + C_1\frac{\text{count}(e_3)}{\sum_e \text{count}(e)} + C_0. \quad (2.9)$$

Here C_n is a smoothing coefficient. Note that the last smoothing coefficient C_0 is used to assure no trigram will get zero.

Language models are based on monolingual corpora which are available in large quantities. The web is a great source for such corpora. Linguistic Data Consortium (LDC) also provides corpora of several billion words in specific languages. Franz and Brants (2006) at Google contribute to the research field a massive 5-gram model trained on one trillion English words from public web pages. It is registered with LDC Catalog number (LDC2006T13). Having such a large model will improve the translation but this advantage is a challenge for the decoder which solves (argmax) function in (2.1) equation (see section 2.4 for more details). Clever methods also should be implemented to fit this model into the memory. Brants et al. (2007) suggests to distribute such a large model over a cluster of machines. Alternatively, Federico and Cettolo (2007) proposed a method of storing the language model on disk using memory mapping.

2.3 Translation Modelling

Statistical machine translation is summarised by the equation (2.3). It states that the most likely translation of a given sentence $\mathbf{f}$ is the sentence that maximises the product of language model $p(\mathbf{e})$ and translation model $p(\mathbf{f}|\mathbf{e})$. Language model assures fluent output while translation model assures adequate meaning. During the search for $\mathbf{e}$, language model estimates the chance that $\mathbf{e}$ is a readable sentence while translation model estimates the chance that $\mathbf{f}$ sentence is the translation of $\mathbf{e}$ sentence. Note that translation model is in the opposite direction of the overall translation system.

A probabilistic translation model is based on a collection of pairs of translated sentences. Usually there are multiple translations for a given text. Collecting statistics from a large parallel corpus helps to find out the most likely translation. Therefore, a common translation in the corpus should have high probability while a rare one has low probability.

In language modelling section, breaking the sentences into smaller parts enables us to collect sufficient statistics. The same approach will be applied in translation modelling. Parallel corpora mostly are sentence-aligned but not word-aligned. This means we know sentence A is translation of sentence B but we do not know the words mapping between each other. However, the alignment is necessary in order to collect the statistics. The problem will be discussed later in section 2.3.1.1.

Translation models are generally divided into three types: word-based, phrase-based and hierarchical phrase-based. The following sections discuss each type and we will start with the simplest one.

2.3.1 Word-based Models

As mentioned earlier, the original work on SMT by Brown et al. (1993) introduced five translation models known as IBM models. These models are based on lexical statistics where words are isolated from their context. Since translation model, $p(\mathbf{f}|\mathbf{e})$, is in the opposite direction of the translation system, $p(\mathbf{e}|\mathbf{f})$, due to applying Bayes rule in equation (2.2), IBM models reason about the probability of a French word is being generated from an English word.

IBM models assume an English word can generate many French words, which means the alignment is one to many. Some English words get dropped and generate no word when a French sentence is shorter than an English sentence. In many cases, there are French words but no English word is responsible for them. Hence, IBM models added NULL token in each English sentence to tackle this issue.

Given a pair of sentences, there are many ways to align the source words to the target words. Therefore, the likelihood of $(\mathbf{f}|\mathbf{e})$ can be written as a mixture of alignment distributions as follows:

$$\begin{aligned} p(\mathbf{f}|\mathbf{e}) &= \sum_{\mathbf{a}} p(\mathbf{f}, \mathbf{a}|\mathbf{e}) \\ &= \sum_{\mathbf{a}} p(J|\mathbf{e}) \prod_{j=1}^{J} p(a_j|a_1^{j-1}, f_1^{j-1}, J, \mathbf{e})\, p(f_j|a_1^{j}, f_1^{j-1}, J, \mathbf{e}), \end{aligned} \tag{2.10}$$

where $\mathbf{a}$ is an alignment vector. The rest of the section explains how each IBM model defines the joint likelihood $p(\mathbf{f}, \mathbf{a}|\mathbf{e})$ although some details will be neglected.

IBM Model 1

IBM Model 1 is a lexical translation model simplifies the alignment problem by assuming that each target position is equally likely to be connected to a source position. This means the word order in both $\mathbf{f}$ and $\mathbf{e}$ is disregarded. The model also assumes that $p(f_j|a_1^j, f_1^{j-1}, J, \mathbf{e})$ depends on f_j and e_{a_j}; and that $p(J|\mathbf{e})$ is independent of J and $\mathbf{e}$.

$$\begin{aligned} \epsilon &\equiv p(J|\mathbf{e}) \\ p(a_j|I) &\equiv p(a_j|a_1^{j-1}, f_1^{j-1}, J, \mathbf{e}) \\ p(f_j|e_{a_j}) &\equiv p(f_j|a_1^{j}, f_1^{j-1}, J, \mathbf{e}) \end{aligned} \tag{2.11}$$

The joint likelihood of $(\mathbf{f}, \mathbf{a}|\mathbf{e})$ in the model is defined as follows:

$$\begin{aligned} p(\mathbf{f}, \mathbf{a}|\mathbf{e}) &= \epsilon \prod_{j=1}^{J} p(a_j|I) p(f_j|e_{a_j}) \\ &= \prod_{j=1}^{J} \frac{\epsilon}{(I+1)} p(f_j|e_{a_j}) \\ &= \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} p(f_j|e_{a_j}). \end{aligned} \tag{2.12}$$

Note that $p(a_j|j) = 1/(I+1)$ for all target positions as assumed. The added one is because the inserted NULL word in the target sentence.

Assuming the parallel corpus is word-aligned for simplicity's sake, the alignment function a is known. The model now builds lexical translation table with probability distribution by relative frequency method,

$$p(f|e) = \frac{\text{count}(f, e)}{\sum_{\hat{f}} \text{count}(\hat{f}, e)}. \tag{2.13}$$

Table 2.1 is an example for a probability distribution of Arabic-English corpus. Arabic is written by Buckwalter transliteration (Buckwalter, 2004).

h*A		Alfndq		yHwy		grf		Sgyrp	
e	$p(f\|e)$	e	$p(f\|e)$	e	$p(f\|e)$	e	$p(f\|e)$	e	$p(f\|e)$
this	0.50	hotel	0.60	contains	0.30	rooms	0.50	small	0.40
that	0.25	motel	0.24	contain	0.30	suites	0.20	little	0.40
the	0.20	house	0.13	has	0.20	beds	0.18	petty	0.15
those	0.04	place	0.02	have	0.10	flats	0.03	minor	0.03
a	0.01	home	0.01	comprise	0.10	space	0.01	short	0.02

TABLE 2.1: Lexical translation probabilities for five Arabic words.

An Arabic sentence such as: "h*A Alfndq yHwy grf Sgyrp" could be generated from various English sentences. The following sentences are some translations with their probabilities according to the model.

Sentence 1: this hotel contains rooms small
$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = 0.50 * 0.60 * 0.30 * 0.50 * 0.40 = 0.018$
Sentence 2: this hotel contains small rooms
$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = 0.50 * 0.60 * 0.30 * 0.40 * 0.50 = 0.018$
Sentence 3: this hotel has small rooms
$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = 0.50 * 0.60 * 0.20 * 0.50 * 0.40 = 0.012$
Sentence 4: that motel have little suites
$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = 0.25 * 0.14 * 0.10 * 0.40 * 0.20 = 0.00028$
Sentence 5: the house comprise petty beds
$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = 0.20 * 0.10 * 0.10 * 0.15 * 0.18 = 0.000054$

Note that the probability of the Arabic sentence being generated from sentence one or sentence two is the same. The model leaves the reordering task to the language model. Sentence three is given less probability but might be preferred by the language model. Last two sentences contain grammatical mistakes. This happens because lexical probabilities in translation models were estimated out of context. This problem could be solved again by the language model.

IBM Model 2

IBM model 1 is a weak lexical translation model. One of its main weaknesses is ignoring the word order. The probability of such a translation with any word order is the same according to model 1 as shown previously. Therefore, IBM model 2 emphasises on reordering by adding absolute alignment model based on the positions of the source and the target words. It is modelled by a probability distribution as

$$p(a_j|j, J, I) \equiv p(a_j|a_1^{j-1}, f_1^{j-1}, J, \mathbf{e}), \tag{2.14}$$

where a_j is a target word position given a source word position j and lengths of source and target sentences.

The translation under model 2 can be seen as a two-step process. First step is lexical translation $p(f_j|e_{a_j})$ as in IBM model 1 and the second one is an alignment step $p(a_j|j, J, I)$. These two steps can be combined in one formula as:

$$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \epsilon \prod_{j=1}^{J} p(f_j|e_{a_j}) p(a_j|j, J, I). \tag{2.15}$$

IBM Model 3

IBM Model 2 is a strong model but not powerful enough. It introduced explicit model for words reordering besides the lexical translation model. However there is more beneficial information in the parallel corpus that can be modelled and it is not considered in Equation (2.10). For example, in a pair of Arabic-English sentences, number of Arabic words could be aligned to an English word. The number is called the fertility of a target word and denoted by ϕ_i. This means a single English word can be a translation of an Arabic phrase, word, or even has no translation such as the word "is" when translated to Arabic. Hence, modelling the fertility of target words is very helpful during translation.

$$p(\phi_i|e_i) \tag{2.16}$$

The fertility model (2.16) is a probability distribution that indicates the probability of how many words $\phi_i = 0, 1, 2, ...$ are generated from a given word e_i. For example, the word "bedroom" will be aligned to two Arabic words "grf nwm", so the fertility model should give a high probability for $\phi_i = 2$ as below.

$$p(2|\text{bedroom}) \simeq 1 \tag{2.17}$$

Note that fertility model allows $\phi_i = 0$ which means the word has no translation in the source language. On the other hand, some source words (Arabic) have no correspondence in the target language (English). As discussed before, IBM models add NULL token for each output sentence to solve this problem.

IBM model 3 combines three components: fertility, lexical translation and distortion (instead of alignment). Distortion model is the opposite of an alignment model. It predicts a source word position j while alignment model predicts a target word position a_j. The mathematical formulation of the model is written as follows:

$$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = p(\phi_0|\phi_1^I, \mathbf{e}) \prod_{i=1}^{I} \phi_i!\; p(\phi_i|e_i) \prod_{j=1}^{J} p(f_j|e_{a_j})\, p(j|a_j, J, I). \tag{2.18}$$

IBM Model 4

In the translation process, phrases tend to move together. Words next to each other in the source sentence are found together in the target sentence. However, in IBM model 3, the distortion component moves words independently. This model improves word reordering by introducing a relative distortion. For more details see sections 3 and 4.6 in Brown et al. (1993).

IBM Model 5

Although IBM model 3 and 4 are advanced translation but they are deficient models. This deficiency is represented by allowing several words to be placed in the same position. Model 5 eliminates deficiency by keeping track of numbers of vacant positions and allowing placement only into these positions. The distortion model is similar to IBM Model 4, except it is based on vacancies. Again for more details see section 4.7 in Brown et al. (1993).

Model 5 is a very complex and addresses many issues in translation but it is very hard to train (Koehn, 2010a). Although the model is no longer the state of the art in translation modelling, it is still the state of the art in the word alignment.

2.3.1.1 Learning IBM Models

In most parallel corpora, sentences are aligned to their translations but there is no word mapping between each other. For each source word f_j, we do not know which of the words in the target sentence $\mathbf{e}$ is its translation. In other words, we lack the alignment function a for this data which is an essential function for parameter estimation in IBM models.

This is a typical problem for machine learning where a model is estimated from incomplete data. Since the alignment between words in the data is hidden from plain view, it is considered a hidden variable in IBM models. If there are word alignments marked up in such data, it would be trivial to do estimation for the lexical translation model. It is simply by collecting counts and performing maximum likelihood estimation as in (2.13). On the other hand, if there is a model given, it would be possible to estimate the most likely alignments between words for each sentence pair. In other words: the model fills in the gap in the data and if there is completed data the model can be estimated.

Alignments

For each pair of sentences there are many possible alignments. If a pair has a single correct alignment, counts could be collected directly. Two equally good-looking alignments means counts should be collected from each one and given half their weights. It is called fractional counts. The following is an example of two sentences ("b c" and "x y") with four possible alignments weighted differently:

$b \rightarrow x$ and $c \rightarrow y$, $p(\mathbf{a}|\mathbf{e},\mathbf{f}) = 0.4$
$b \rightarrow y$ and $c \rightarrow x$, $p(\mathbf{a}|\mathbf{e},\mathbf{f}) = 0.3$
$b \rightarrow x, y$ and $c \rightarrow \varnothing$, $p(\mathbf{a}|\mathbf{e},\mathbf{f}) = 0.1$
$b \rightarrow \varnothing$ and $c \rightarrow x, y$, $p(\mathbf{a}|\mathbf{e},\mathbf{f}) = 0.2$

For example, in order to estimate the fertility $p(1|b)$ (the chance that word b aligned only to one word), we will count how many times this occurred and divide it by all possible fertilities. Obviously, the first two alignments are mapping b to one word.

$$\begin{aligned} p(1|b) &= \frac{\text{count}(1|b)}{\sum_i \text{count}(i|b)} \\ &= \frac{\text{count}(1|b)}{\text{count}(0|b) + \text{count}(1|b) + \text{count}(2|b)} \\ &= \frac{(0.4 + 0.3)}{(0.2) + (0.4 + 0.3) + (0.1)} = 0.7 \end{aligned} \tag{2.19}$$

The main goal of IBM models is to estimate translation model $p(\mathbf{f}|\mathbf{e})$ from a parallel corpus. Sentence $\mathbf{f}$ can be produced from sentence $\mathbf{e}$ in many ways. The models estimate the probability of sentence $\mathbf{f}$ with a particular alignment $p(\mathbf{f}, \mathbf{a}|\mathbf{e})$. Therefore, the probability of translation model written as follows:

$$p(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} p(\mathbf{f}, \mathbf{a}|\mathbf{e}) \tag{2.20}$$

Alignment probability, $p(\mathbf{a}|\mathbf{e}, \mathbf{f})$, leads to estimate parameter values (fertility, lexical translation and distortion) in IBM models. Since the data is incomplete (no alignments), we need to compute $p(\mathbf{a}|\mathbf{e}, \mathbf{f})$. Applying the chain rule ($p(x, y) = p(x).p(y|x)$), gives us:

$$\begin{aligned} p(\mathbf{a}|\mathbf{e}, \mathbf{f}) &= \frac{p(\mathbf{a}, \mathbf{e}, \mathbf{f})}{p(\mathbf{e}, \mathbf{f})} \\ &= \frac{p(\mathbf{e}).p(\mathbf{f}, \mathbf{a}|\mathbf{e})}{p(\mathbf{e}).p(\mathbf{f}|\mathbf{e})} \\ &= \frac{p(\mathbf{f}, \mathbf{a}|\mathbf{e})}{p(\mathbf{f}|\mathbf{e})} \\ &= \frac{p(\mathbf{f}, \mathbf{a}|\mathbf{e})}{\sum_{\mathbf{a}} p(\mathbf{f}, \mathbf{a}|\mathbf{e})} \end{aligned} \tag{2.21}$$

It is clear that computing alignment probabilities $p(\mathbf{a}|\mathbf{e}, \mathbf{f})$ is based on IBM models. The models also need alignment probabilities for parameter estimation. This is known as a chicken and egg problem.

Expectation Maximisation Algorithm

The expectation maximization (EM) algorithm addresses the situation of incomplete data. It is an iterative learning method that fills in the gaps in the data and trains a

model in alternating steps. Knight (1997) in his article (page 84) gave a good example of how to build alignments manually for unaligned sentence pairs by gathering information incrementally. In Table 2.2, we give a short example for building alignments intuitively.

Parallel English-Swedish sentences:

I love you ||| Jag älskar dig

I want to help him ||| Jag vill hjälpa honom

I do not want him ||| Jag vill inte ha honom

We love him ||| Vi älskar honom

Step 1: I = Jag , him = honom (reason: repeated 3 times together)
Step 2: love = älskar (reason: repeated in the first and last sentences)
Step 3: you = dig , We = Vi (reason: the remaining words after step 1 and 2)
Step 4: want = vill (reason: unknown word repeated in 2nd and 3rd sentences)
Step 5: to help = hjälpa , do not = inte ha (reason: the remaining words)

TABLE 2.2: An example for building alignments manually from parallel sentences

The EM algorithm in general works as following:

1) Initialize the model, typically with uniform distributions.
2) Apply the model to the data (expectation step).
3) Learn the model from the data (maximisation step).
4) Iterate second step and third one until convergence.

Knight (1999) explained clearly how EM algorithm works in the context of statistical machine translation. Therefore, we reproduce it here and expand the calculations for more clarification.

He assumes there is a very small parallel corpus with just two pairs of sentences: first pair is (**e**:"b c" , **f**:"x y") and second pair is (**e**:"b" , **f**:"y"). For simplicity, first pair has only two possible alignments: ($b \rightarrow x$, $c \rightarrow y$) or ($b \rightarrow y$, $c \rightarrow x$). Second pair has one alignment ($b \rightarrow y$). Hence, the estimation is just for lexical translation parameter (i.e. IBM model 1). Table 2.3 shows how we train IBM mode 1 using EM algorithm.

It is clear that EM algorithm selects the alignment ($b \rightarrow y$, $c \rightarrow x$) for the first sentence pair. This happened because the second pair only has one alignment ($b \rightarrow y$). This information helps to increase $p(y|b)$ and downgrade $p(y|c)$. This example was EM for IBM model 1, however EM for higher models is similar except that parameter estimation is more time consuming because there are more parameters need to be estimated each iteration (fertility and distortion).

CORPUS: pair 1 = (**e**:"b c" , **f**:"x y") , pair 2 = (**e**:"b" , **f**:"y") , fertility(ϕ)=1

Step 1: Uniform distribution
$p(x|b) = 0.5 \qquad p(y|b) = 0.5 \qquad p(x|c) = 0.5 \qquad p(y|c) = 0.5$

Step 2: Expectation
(A) Compute $p(\mathbf{f}, \mathbf{a}|\mathbf{e})$ for all alignments
$(b \to x,\ c \to y)$, $p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = p(x|b) * p(y|c) = 0.5 * 0.5 = 0.25$
$(b \to y,\ c \to x)$, $p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = p(y|b) * p(x|c) = 0.5 * 0.5 = 0.25$
$(b \to y)$, $p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = p(y|b) = 0.5$

(B) Normalise $p(\mathbf{f}, \mathbf{a}|\mathbf{e})$ values to yield $p(\mathbf{a}|\mathbf{e}, \mathbf{f})$ values: $p(\mathbf{a}|\mathbf{e}, \mathbf{f}) = \frac{p(\mathbf{f},\mathbf{a}|\mathbf{e})}{\sum_a p(\mathbf{f},\mathbf{a}|\mathbf{e})}$
$(b \to x,\ c \to y)$, $p(\mathbf{a}|\mathbf{e}, \mathbf{f}) = \frac{0.25}{0.25+0.25} = 0.5$
$(b \to y,\ c \to x)$, $p(\mathbf{a}|\mathbf{e}, \mathbf{f}) = \frac{0.25}{0.25+0.25} = 0.5$
$(b \to y)$, $p(\mathbf{a}|\mathbf{e}, \mathbf{f}) = \frac{0.5}{0.5} = 1$: it is 1 because there is only one alignment.

(C) Collect fractional counts to yield the expectation of count(f, e)
count$(x, b) = 0.5$, count$(y, b) = 0.5 + 1 = 1.5$, count$(x, c) = 0.5$, count$(y, c) = 0.5$

Step 3: Maximisation
Normalise fractional counts to get revised parameter values: $p(f|e) = \frac{\text{count}(f,e)}{\sum_{\hat{f}} \text{count}(\hat{f},e)}$
$p(x|b) = \frac{\text{count}(x,b)}{\text{count}(x,b)+\text{count}(y,b)} = \frac{0.5}{0.5+1.5} = 0.25$
$p(y|b) = \frac{\text{count}(y,b)}{\text{count}(x,b)+\text{count}(y,b)} = \frac{1.5}{0.5+1.5} = 0.75$
$p(x|c) = \frac{\text{count}(x,c)}{\text{count}(x,c)+\text{count}(y,c)} = \frac{0.5}{0.5+0.5} = 0.5$
$p(y|c) = \frac{\text{count}(y,c)}{\text{count}(x,c)+\text{count}(y,c)} = \frac{0.5}{0.5+0.5} = 0.5$

Step 4: Repeat step 2 and step 3 until convergence
$p(x|b) = 0.0001 \qquad p(y|b) = 0.9999 \qquad p(x|c) = 0.9999 \qquad p(y|c) = 0.0001$

TABLE 2.3: Learning IBM model 1 by EM algorithm [summarised from Knight (1999)]

Efficient Learning

In the previous section, EM algorithm was applied to limited alignment assuming the fertility is always one. The NULL token was also ignored. However, possible alignments for sentence pair with two words length is not just two. In fact, there are nine alignments as shown below. Note that word e_0 represents NULL token in **e** sentence.

$$(e_0 \to f_1\ ,\ e_0 \to f_2)\ ,\ (e_0 \to f_1\ ,\ e_1 \to f_2)\ ,\ (e_0 \to f_1\ ,\ e_2 \to f_2)$$
$$(e_1 \to f_1\ ,\ e_0 \to f_2)\ ,\ (e_1 \to f_1\ ,\ e_1 \to f_2)\ ,\ (e_1 \to f_1\ ,\ e_2 \to f_2)$$
$$(e_2 \to f_1\ ,\ e_0 \to f_2)\ ,\ (e_2 \to f_1\ ,\ e_1 \to f_2)\ ,\ (e_2 \to f_1\ ,\ e_2 \to f_2)$$

In general, there are $(I+1)^J$ possible alignments (1 is for NULL). This means normalising $p(\mathbf{a}|\mathbf{e}, \mathbf{f})$ in EM algorithm requires $J * (I + 1)^J$ operations as in Equation (2.20). The computational complexity is exponential. This problem can be simplified into quadratic

if we take advantage of certain regularities. In the example above, there is a common part in each line of the alignments. This leads to compute $p(\mathbf{f}|\mathbf{e})$ according to IBM model 1 efficiently as shown in Equation (2.22). The operations were reduced into $(I+1) \times J$ and that the significance of the last step. The computational complexity in general is roughly $O(n^2)$ since $I \sim J$.

$$
\begin{aligned}
p(\mathbf{f}|\mathbf{e}) &= \sum_{a} p(\mathbf{f}, \mathbf{a}, |\mathbf{e}) \\
&= \sum_{a} \prod_{j=1}^{J} p(f_j|e_{a_j}) \\
&= p(f_1|e_0) \times p(f_2|e_0) + p(f_1|e_0) \times p(f_2|e_1) + p(f_1|e_0) \times p(f_2|e_2) \\
&+ p(f_1|e_1) \times p(f_2|e_0) + p(f_1|e_1) \times p(f_2|e_1) + p(f_1|e_1) \times p(f_2|e_2) \\
&+ p(f_1|e_2) \times p(f_2|e_0) + p(f_1|e_2) \times p(f_2|e_1) + p(f_1|e_2) \times p(f_2|e_2) \\
\\
&= p(f_1|e_0) \times [p(f_2|e_0) + p(f_2|e_1) + p(f_2|e_2)] \\
&+ p(f_1|e_1) \times [p(f_2|e_0) + p(f_2|e_1) + p(f_2|e_2)] \\
&+ p(f_1|e_2) \times [p(f_2|e_0) + p(f_2|e_1) + p(f_2|e_2)] \\
&= [p(f_1|e_0) + p(f_1|e_1) + p(f_1|e_2)] \times [p(f_2|e_0) + p(f_2|e_1) + p(f_2|e_2)] \\
&= \prod_{j=1}^{J} \sum_{i=0}^{I} p(f_j|e_i)
\end{aligned}
\tag{2.22}
$$

Learning IBM model 1 is fast because it has only lexical translation parameters. Higher models have more parameters to estimate which requires more computation each iteration in EM algorithm. Instead of initialise their parameters with uniform distribution. We can transfer parameter values from a lower model after convergence to the higher one (Knight, 1999). We implemented model 1 and 2 in MATLAB (see Appendix B.5 and B.6).

2.3.1.2 Supervised Learning

Machine translation based on a manually word-aligned corpus works better than aligned one with unsupervised approaches such as IBM models (Callison-Burch et al., 2004). However, manually word-aligned corpora do not exist in large quantities and manual alignment process is expensive and time consuming.

Supervised machine learning techniques have been suggested by many researchers to formulate word alignment as structured learning problem. Since most available data are unaligned, discriminative approaches used over a baseline unsupervised model (e.g. IBM models). Model's parameters are then optimised during machine learning over small aligned data.

In the recent past years, many discriminative approaches were proposed. Moore (2005) and Moore et al. (2006) used perceptron in discriminative framework for word alignment. Ittycheriah and Roukos (2005) propose maximum entropy models. Ayan et al. (2005) combined word alignments using neural networks. Taskar et al. (2005) present large-margin approach to feature-based matching. Boosting method was suggested to improve statistical word alignment (Wu and Wang, 2005; Wu et al., 2006). Cherry and Lin (2006) add soft Syntactic constraints for word alignment through support vector machines. Blunsom and Cohn (2006) and also Niehues and Vogel (2008) present conditional random fields. Venkatapathy and Joshi (2007) used the structured learning method MIRA to reduce the alignment errors especially for language pairs with high structural divergence.

2.3.2 Phrase-based Models

IBM models are referred to as word-based models. While these early models assume the translation to take place on a word by word basis, in reality, groups of words (phrases) are recognised as better units of translation (Koehn, 2010a). In many cases, phrases are translated as a whole. Then, they might be reordered. Phrase translation has many advantages over word translation. The models are conceptually simpler. Besides that, they learn local reorderings and translations of phrase expressions. Translating phrases instead of words helps to resolve many ambiguities.

Phrase-based models are built from word alignments generated by IBM models. As mentioned in the previous section, EM algorithm fills in the gap in our data by iterating over parallel sentences to solve the alignment between their words. Word alignment based on IBM models is a one-to-many relation while the phrases and their translations could be more than one word. Hence, we need many-to-many relation.

The solution is to train IBM models on both directions (source to target and target to source) then combining word alignments as shown in Figure 2.1. Och and Ney (2003) introduced different methods for symmetrising word alignments.

Phrase Extraction

The goal of building word alignments for each sentence pair is to create a translation table from the whole corpus. We do that by extracting phrase pairs that are consistent with the word alignments. Figure 2.2 illustrates what kind of phrase pairs are included and excluded. Table 2.4 displays the complete list of extracted phrase pairs from the union word alignment in Figure 2.1.

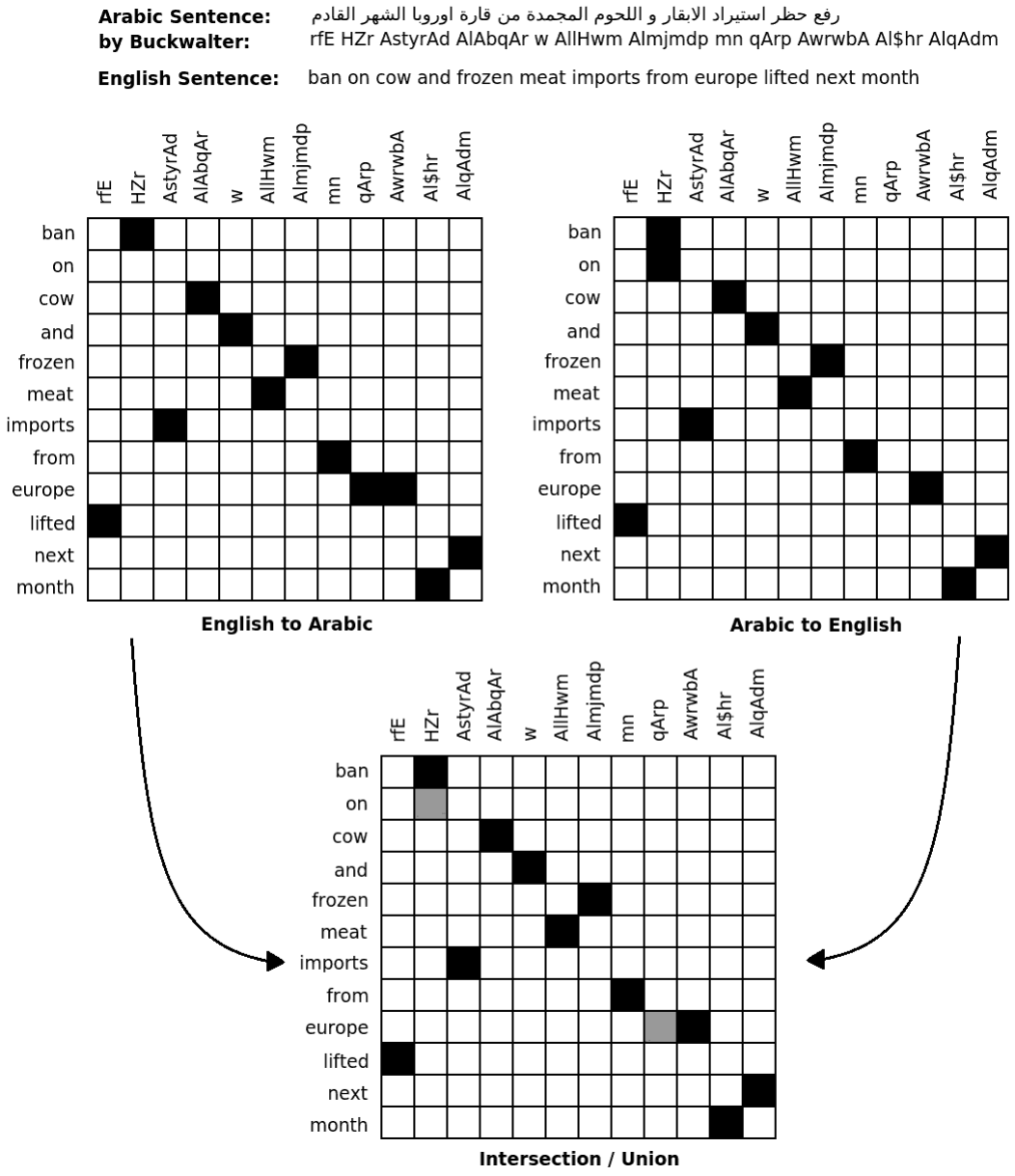

FIGURE 2.1: Symmetrisation of word alignments for English sentence and Arabic one

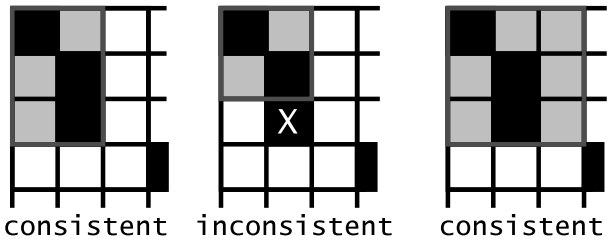

FIGURE 2.2: Definition of phrase pairs being consistent with a word alignment

ban on — HZr
ban on cow and frozen meat imports — HZr AstyrAd AlAbqAr w AllHwm Almjmdp
ban on cow and frozen meat imports from
— HZr AstyrAd AlAbqAr w AllHwm Almjmdp mn
ban on cow and frozen meat imports from europe
— HZr AstyrAd AlAbqAr w AllHwm Almjmdp mn qArp AwrwbA
ban on cow and frozen meat imports from europe lifted
— rfE HZr AstyrAd AlAbqAr w AllHwm Almjmdp mn qArp AwrwbA
ban on cow and frozen meat imports from europe lifted next month
— rfE HZr AstyrAd AlAbqAr w AllHwm Almjmdp mn qArp AwrwbA Al$hr AlqAdm
cow — AlAbqAr **cow and** — AlAbqAr w
cow and frozen meat — AlAbqAr w AllHwm Almjmdp
cow and frozen meat imports — AstyrAd AlAbqAr w AllHwm Almjmdp
cow and frozen meat imports from — AstyrAd AlAbqAr w AllHwm Almjmdp mn
cow and frozen meat imports from europe
— AstyrAd AlAbqAr w AllHwm Almjmdp mn qArp AwrwbA
and — w
and frozen meat — w AllHwm Almjmdp
frozen — Almjmdp
frozen meat — AllHwm Almjmdp
meat — AllHwm
imports — AstyrAd
from — mn
from europe — mn qArp AwrwbA
lifted — rfE
next — AlqAdm
next month — Al$hr AlqAdm
month — Al$hr

TABLE 2.4: Extracted phrase pairs from the union word alignment in Figure 2.1

Context-based Phrase Estimation

Previously translation probabilities are estimated based on their occurrences out of context. The most likely translation for a phrase within a context is selected by the decoder with the help of other models such as language model. Hence, having a translation model able to explore the context will improve the phrase translation and ease the decoder workload.

Vickrey et al. (2005) propose a word sense disambiguation (WSD) model for word translation based on syntax information. Bangalore et al. (2007) also propose a similar model called global lexical selection model using maximum entropy classifier trained on bag-of-words features. For phrase translation, Carpuat and Wu (2007) combined four WSD models proposed by Vickrey et al. (2005). Giménez and Màrquez (2007) formulate phrase translation as a classification problem using SVM. Ni et al. (2010b) applied max-margin structure (MMS) to model phrase translation using perceptron-based learning algorithm instead of SVM because of its time complexity.

2.3.3 Hierarchical Phrase-based Models

Chiang (2007) realises the ability of phrase-based models to learn word reordering and propose to extend them to learn phrase reordering. The approach capitalises on the strengths of phrase-based models by using hierarchical phrases (i.e. phrases comprise sub-phrases). It is formalised as synchronous context free grammar (SCFG) rules induced from a parallel corpus without syntactic annotations. For example, we can learn the following Arabic-English rule:

$$\mathrm{X} \rightarrow \langle \text{lA yHb } \mathrm{X}_1 \text{ An } \mathrm{X}_2,\ \mathrm{X}_1 \text{ does not like to } \mathrm{X}_2 \rangle, \tag{2.23}$$

where X_1 and X_2 are referred to as non-terminal symbols for sub-phrases and the words are terminal.

SCFG formalism added modelling power which is able to learn long-distance reordering but comes with complexity. Shallow grammars have been proposed in order to limit the complexity proportional to the improvement in translation quality (Iglesias et al., 2009). For example, de Gispert et al. (2010) find that for Arabic-English translation shallow-1 grammars are sufficient. On the other hand, for Chinese-English translation shallow-3 grammars are necessary to produce translation comparable to full hierarchical system (i.e. arbitrary nesting).

Rule Extraction

In hierarchical phrase-based models, there are three types of SCFG rules: phrasal, hierarchical and glue. Phrasal rules consist of consecutive terminal symbols (only words). All phrase pairs as in Table 2.4 form phrasal rules. Hierarchical rules consist of terminal and non-terminal symbols (words and variables). These rules learn phrase reordering as part the translation model and are also extracted from word alignment as shown in Figure 2.3. In Table 2.5, we show some extracted rules. Glue rules consist of non-terminal symbols (only variables) and begin with S (the start symbol) as follows:

$$\mathrm{S} \rightarrow \langle \mathrm{S}\ \mathrm{X}, \mathrm{S}\ \mathrm{X} \rangle \tag{2.24}$$

$$\mathrm{S} \rightarrow \langle \mathrm{X}, \mathrm{X} \rangle. \tag{2.25}$$

They allow to merge translated phrases when there is no hierarchical rule to be derived. In fact, using only phrasal and glue rules reduce the hierarchical system to phrase-based one with monotone translation (i.e. no reordering).

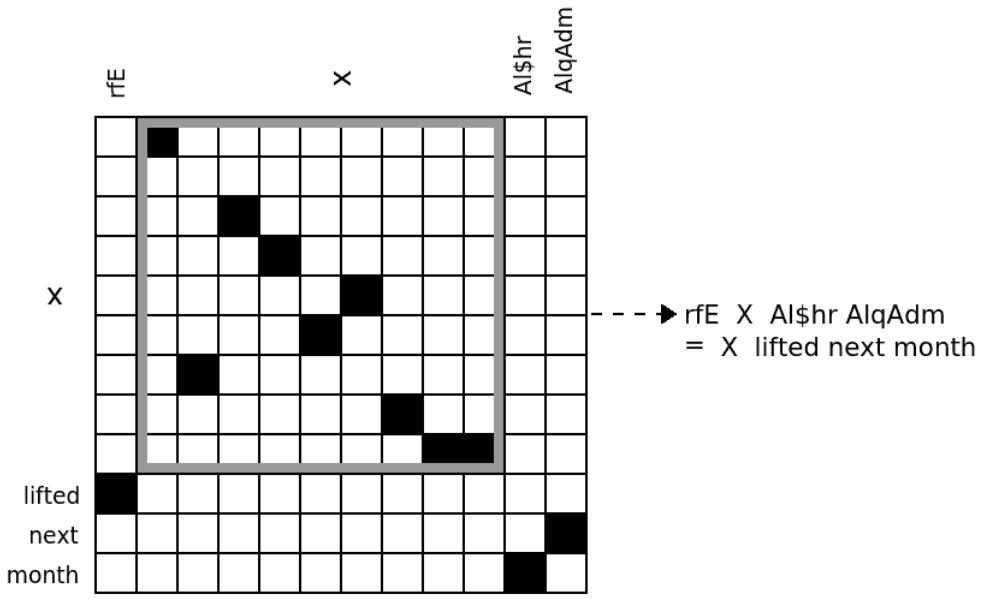

FIGURE 2.3: Extracting hierarchical phrase rules from word alignment

Phrasal Rules:
X → ⟨HZr, ban on⟩
X → ⟨AlAbqAr, cow⟩
.
.
Hierarchical Rules:
X → ⟨rfE X, lifted X⟩
X → ⟨AstyrAd X mn, X imports from⟩
X → ⟨HZr AstyrAd X, ban on X imports⟩
X → ⟨X mn qArp AwrwbA, X europe⟩
X → ⟨rfE X_1 mn X_2, X_1 from X_2 lifted⟩
X → ⟨rfE X Al$hr AlqAdm, X lifted next month⟩
X → ⟨rfE HZr X Al$hr AlqAdm, ban on X lifted next month⟩
.
.

TABLE 2.5: Extracted grammar rules from the union word alignment in Figure 2.1

2.4 Decoding

The most likely translation of a given sentence **f** is by seeking a translation **e** that maximises the product of language model $p(\mathbf{e})$ and translation model $p(\mathbf{f}|\mathbf{e})$. This process (decoding) is very hard because there are an exponential number of choices. Figure 2.4 shows the amount of choices for translating an Arabic sentence into English with limited phrase translations. Exhaustive search for all options is very expensive particularly with long sentences.

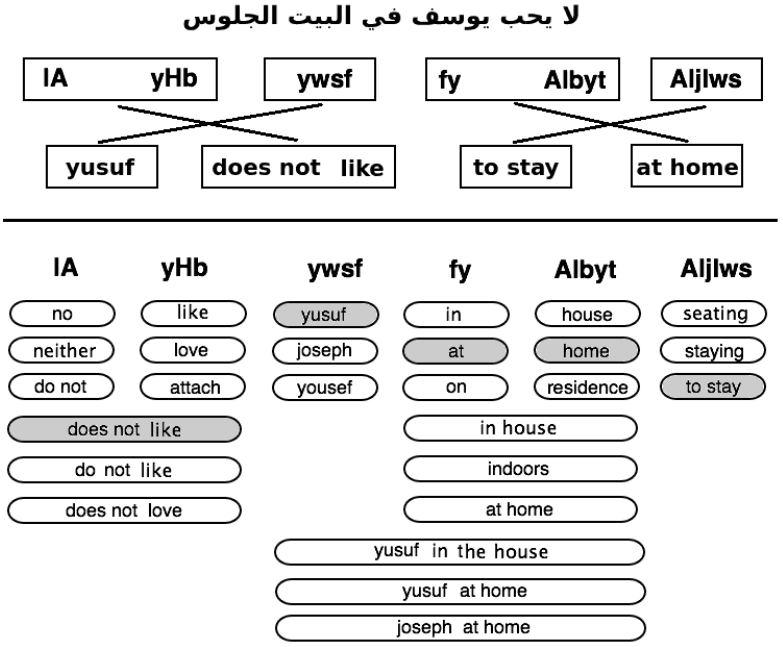

FIGURE 2.4: Arabic sentence with three translations for each phrase

Efficient search algorithms are vital in machine translation. Many algorithms were proposed for such a problem such as Beam Search and Greedy hill-climbing (Koehn, 2010a). They use heuristic methods which means there is no guarantee of an optimal solution.

A decoding algorithm during its search constructs partial translations called hypotheses. Figure 2.5 shows the decoding process for the Arabic sentence "lA yHb ywsf fy Albyt Aljlws". It starts with an empty hypothesis (no word translated). Then, it selects one of the options in Figure 2.4. Hypotheses are expanded until all words are covered. The best hypothesis is the one with the highest probability according the language and translation models.

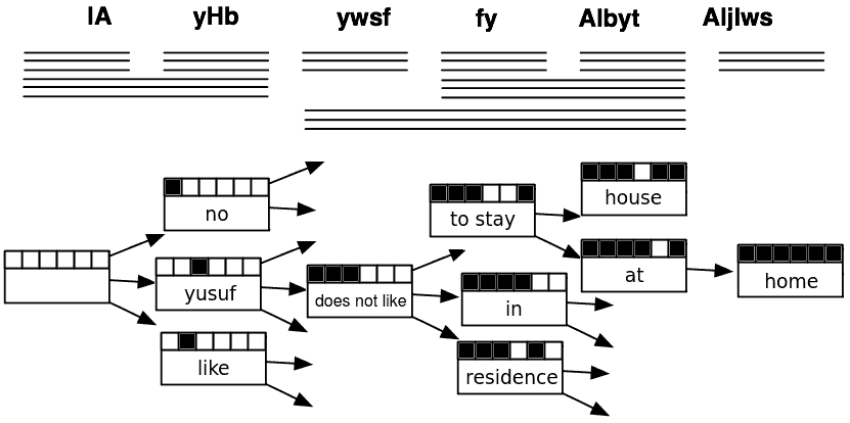

FIGURE 2.5: Decoding process for an Arabic sentence

In hierarchical phrase-based systems, hypotheses are no longer constructed from left to right as in Figure 2.5. Grammar rules impose decoding with gaps due to non-terminal symbols. Since context-free grammar (CFG) parsing is similar to synchronous CFG decoding, many decoders adopt popular parsing algorithms (Chiang, 2007). They grow a synchronous tree (bottom to top) to cover each time longer spans of the source sentence as shown in Figure 2.6. In practice, SCFG decoding is slower than the traditional phrase-based decoding with limited reordering (Lopez, 2008).

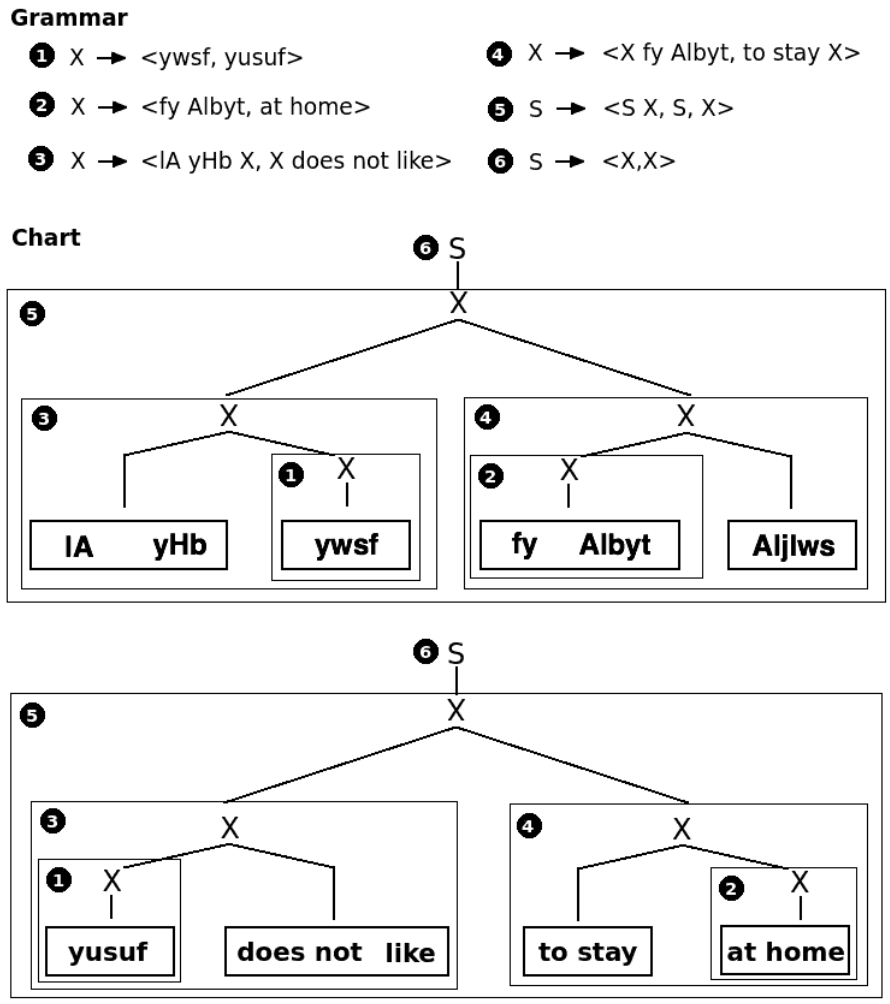

FIGURE 2.6: Extracting hierarchical phrase rules from word alignment

There are many ways to cut the computational cost of the decoding process. For example, some hypotheses produce the same translation and recombining them limit the search space. In addition to that, hypotheses with low probability can be pruned early. However, pruning is sometimes very risky because hypotheses that started with the easy parts of a sentence get higher score than hypotheses started with the difficult parts (Koehn, 2010a).

2.5 Evaluation

In translation, there is no unique answer. A sentence has many acceptable translations. The quality of these translations are subjective to humans' judgement which means evaluating machine translation systems is difficult. The following Chinese sentence is a popular example used by many researchers taken from the 2001 NIST evaluation set (first NIST set). Note there are 10 acceptable translations.

这个 机场 的 安全 工作 由 以色列 方面 负责 .

Israeli officials are responsible for airport security.
Israel is in charge of the security at this airport.
The security work for this airport is the responsibility of the Israel government.
Israeli side was in charge of the security of this airport.
Israel is responsible for the airport's security.
Israel is responsible for safety work at this airport.
Israel presides over the security of the airport.
Israel took charge of the airport security.
The safety of this airport is taken charge of by Israel.
This airport's security is the responsibility of the Israeli security officials.

Evaluation criteria are adequacy (translation conveys the same meaning) and fluency (readable text). Many evaluation metrics have been proposed and they fall in two categories: manual metrics (human's judgement) and automatic metrics. The second one is more practical for evaluating translation systems for many reasons. First, they are not expensive. Second, system's performance can be optimised automatically towards these metrics. Third, repeated use of a metric gives same results (consistent). Finally, they are much faster than humans.

Automatic metrics are based on a set of human translations called references. Precision, Recall and WER are simple automatic metrics. Precision metric is simply the ratio of the correct words to the translation as follows:

$$\text{Precision} = \frac{\text{correct words}}{\text{translation length}}. \tag{2.26}$$

Recall metric is the ratio of recalled words from the reference as follows:

$$\text{Recall} = \frac{\text{correct words}}{\text{reference length}}. \tag{2.27}$$

The major weakness of Precision and Recall is that sentences with different word order are given the same score. WER (Word Error Rate) tries to recover this weakness by

measuring the minimum number of editing steps needed to transform the translation to the reference as follows:

$$\text{WER} = \frac{\text{substitutions} + \text{insertions} + \text{deletions}}{\text{reference length}}. \tag{2.28}$$

Currently the most popular automatic evaluation metric is BLEU (Bilingual Evaluation Understudy). It was introduced by Papineni et al. (2002). The metric computes precision for n-grams and is defined as follows:

$$\text{BLEU} = \text{BP} * \exp \sum_{i=1}^{n} \lambda_i * \log(\text{precision}_\text{i}), \tag{2.29}$$

where BP is a brevity penalty defined as:

$$\text{BP} = \min(1, \frac{\text{translation length}}{\text{reference length}}). \tag{2.30}$$

The brevity penalty reduces BLEU score if the translation is much shorter than the reference. Each n-gram precision is weighted by λ_i. The longest matched n-gram is usually 4 and the weights typically set to 1. This simplifies the formula to the following:

$$\text{BLEU4} = \min(1, \frac{\text{translation length}}{\text{reference length}}) \prod_{\text{i}=1}^{4} \text{precision}_\text{i}. \tag{2.31}$$

Note that if any of the n-gram perceptions is 0 the whole score will be 0. Therefore, BLEU score is usually computed over the entire test sentences.

2.6 Discriminative Training

The state of the art in statistical machine translation has many models other than language and translation models. Although these two models are the main ones in any machine translation system. The overall translation probability $p(\mathbf{e}|\mathbf{f})$ is a combination of several models vary in their importance to the system. Hence, having a log-linear framework enables us to add new models easily as in:

$$p(x) = \exp \sum_{i=1}^{n} \lambda_i h_i(x) \tag{2.32}$$

where each model h_i is weighted by a parameter λ_i.

Koehn et al. (2005) describe a statistical machine translation system using 7 models: language model, phrase translation, lexical translation, word penalty, phrase penalty,

linear reordering penalty and lexicalised reordering. In Moses (open source SMT system), the score function (2.33) contains 5 models which are translation model $p_t(\mathbf{f}|\mathbf{e})$, language model $p_{lm}(\mathbf{e})$, lexical weight translation model $p_{lex}(\mathbf{f}|\mathbf{e})$, phrase reordering model[1] $p_d(\mathbf{f},\mathbf{e})$ and word penalty w.

$$\begin{aligned}\mathbf{e}_{\text{best}} &= \underset{\mathbf{e}}{\operatorname{argmax}}\, p(\mathbf{e}|\mathbf{f}) \\ &= \underset{\mathbf{e}}{\operatorname{argmax}} \left\{ p_t(\mathbf{f}|\mathbf{e})^{\lambda_t} p_{lm}(\mathbf{e})^{\lambda_{lm}} p_{lex}(\mathbf{f}|\mathbf{e})^{\lambda_{lex}} p_d(\mathbf{f},\mathbf{e})^{\lambda_d} w^{|\mathbf{e}|\lambda_w} \right\} \\ &= \underset{\mathbf{e}}{\operatorname{argmax}} \sum_{i\in\{t,lm,lex,d,w\}}^{n} \lambda_i \log p_i(\mathbf{f},\mathbf{e}) \end{aligned} \tag{2.33}$$

As mentioned earlier, statistical models vary in their importance to the system. Log-linear framework gives a weight for each model which affects its contribution to the translation system. Setting these parameters in order to discriminate between bad and good translations is not an easy task. Various methods have been proposed for parameter tuning. For example, Och and Ney (2002) introduce maximum entropy models for statistical machine translation and use Generalised Iterative Scaling (GIS) algorithm (Darroch and Ratcliff, 1972) for training. Shen et al. (2004) suggest discriminative reranking using perceptron-based algorithms.

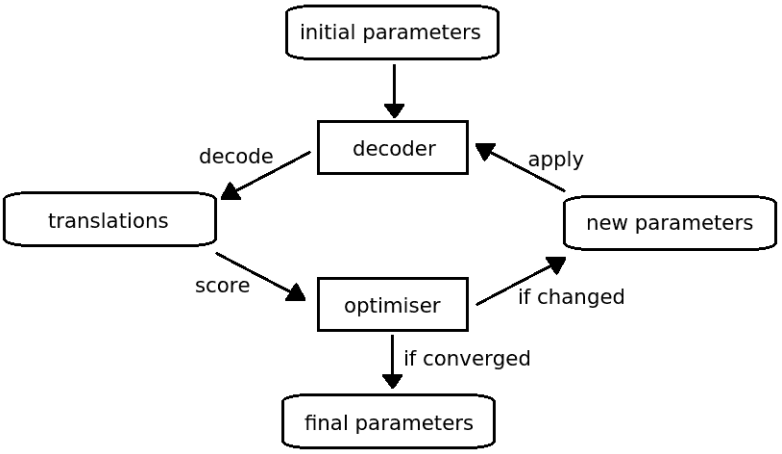

FIGURE 2.7: Iterative parameter tuning

Och (2003) proposes Minimum Error Rate Training (MERT) method to tune the parameters using Powell search (search space reduction technique). Basically, a baseline system is given a set of sentences (development set) to generates an n-best list of candidate translations for each sentence. These candidates are represented by a set of features h_i. Then features' weights λ_i are adjusted to give preference to good translations according to an evaluation metric (e.g BLEU score) as shown in Figure 2.7. MERT is

[1]This thesis is trying to improve this particular model.

computationally expensive method but we do not have to use large data set to find the best weights for our models. Usually, a small development set (e.g. 1000 sentences) is enough. The set should not be part of the training data to avoid overfitting. There are other search techniques can be used instead of Powell search such as simplex algorithm (Koehn, 2010a).

Sparse Features

Previously, only a few features are optimised (dense features). In large-scale discriminative training, we use thousands or millions of features. For instance, instead of learning translation probabilities given a source phrase to maximise the likelihood, we want to learn directly how useful these translations in terms of an evaluation metric (e.g. BLEU). Discriminative models work better than generative models but the bottleneck is that they are very computationally expensive and memory consuming. Therefore, their performance might be worse than generative models if they cannot scale up to handle large data.

Several machine learning methods have been proposed for large-scale discriminative learning. Tillmann and Zhang (2005) suggest a localised prediction model. Liang et al. (2006) propose an end-to-end discriminative approach. Arun and Koehn (2007) introduce online learning methods for discriminative training of phrase-based system. Kernel regression methods also were applied (Wang et al., 2007; Wang and Shawe-Taylor, 2008).

2.7 Overall System Structure

In this thesis, we used a phrase-based system. Figure 2.8, produced by an experimental management system (Koehn, 2010b), illustrates the steps in Moses to build it. Blue boxes mean steps that we have actually run in our experiments. The overall system structure can be summarised in five main phases:

1. A parallel corpus is tokenised, cleaned (e.g. long and short sentences) and trucased.
2. A language model is built from the target side or from a monolingual corpus.
3. Training is the most time consuming phase as the system learns word alignments to extract phrase pairs. Then, a translation and reordering models are built.
4. The system tunes the weight of each feature (e.g. language model). This step is also computationally expensive.
5. Finally, automatic metrics (e.g. BLEU) evaluate the system based on a set of human translations.

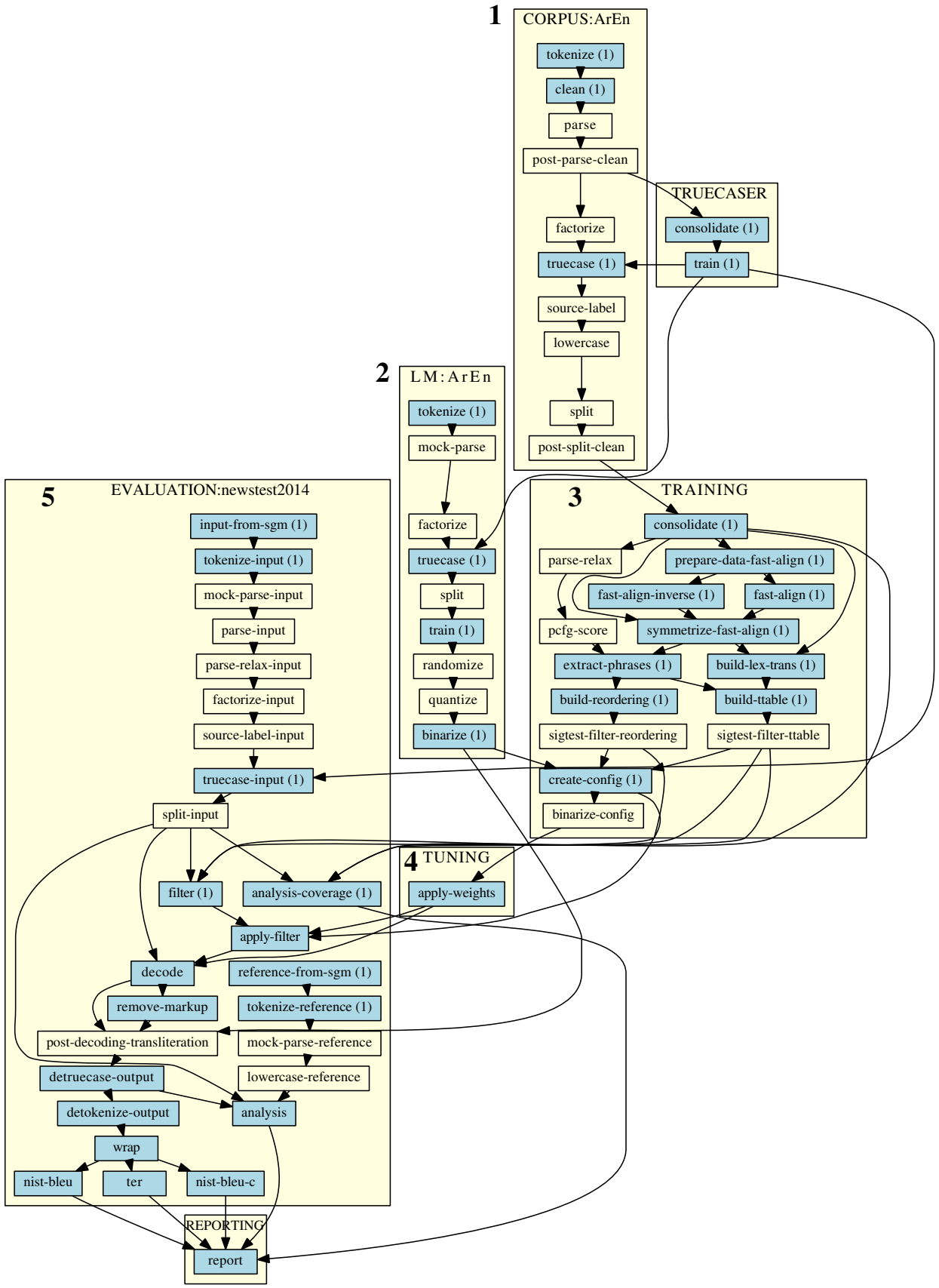

FIGURE 2.8: Steps of building a pharse-based machine translation system.

2.8 Chapter Summary

In this chapter, we covered most of the fundamental concepts of statistical machine translation. Starting from the mathematical basis of SMT, we introduced five statistical models widely known as the IBM models (Brown et al., 1988). While these early models assume the translation to take place on a word by word basis, phrases are recognised as better units. This realisation led to phrase-based models which are conceptually simpler and able to learn local reorderings and resolve many ambiguities. Phrase-based models are built from word alignments generated by IBM models (Och and Ney, 2003). Later, Chiang (2007) further extends these models to learn phrase reordering. The approach capitalises on the strengths of phrase-based models by using hierarchical phrases (i.e. phrases comprise sub-phrases).

Hierarchical systems are able to learn long-distance reordering but comes with complexity. On the other hand, phrase-based systems are faster with less reordering modelling power (Lopez, 2008) which we are trying to improve in this thesis. In the next chapter we have a review of the literature in reordering models.

‘

Chapter 3

Reordering Models

In state-of-the-art phrase-based statistical machine translation systems, modelling phrase reorderings is an important need to enhance naturalness of the translated outputs, particularly when the grammatical structures of the language pairs differ significantly. While phrase-based systems are a significant improvement over word-based approaches, a particular issue that follows is long range reorderings at the phrase level (Galley and Manning, 2008). Figure 3.1 shows a number of required arrangements of English phrases translated from an Arabic sentence in order to have fluent output.

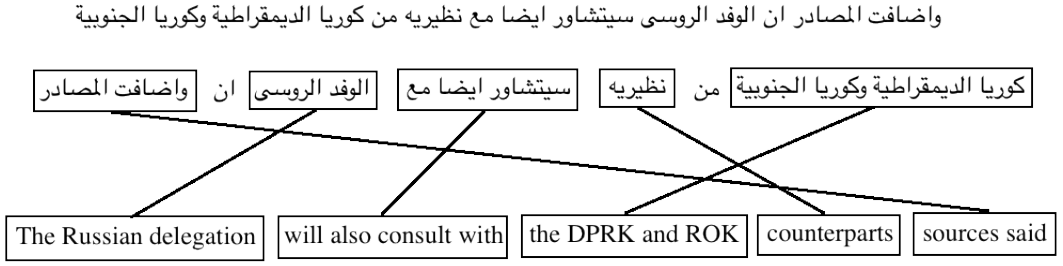

FIGURE 3.1: The problem of phrase reordering is illustrated by an Arabic sentence translated to English, taken from a NIST test set. Note that Arabic is written from right to left but we present it as English for the sake of clarity.

The translation model can capture the local meaning for each source phrase. However, to capture the whole meaning of a sentence, its translated phrases need to be in the correct order. The language model, which ensures fluent translation, plays an important role in phrase reordering; however, it prefers sentences that are grammatically correct without considering their actual meaning (i.e. the dependence of the target sentence on the source sentence). Besides that, it has a bias towards short translations (Koehn, 2010a). Therefore, developing a specific reordering model will improve the accuracy particularly when translating between two grammatically different languages.

Early work on handling phrase reorderings implemented a relaxation into the decoder which, instead of forcing phrases to be in synchrony, allowed a penalty function that

penalised large movements proportionately (Koehn, 2004a). An alternative approach, adopted by several systems nowadays is lexicalised reordering modelling (Tillmann, 2004; Kumar and Byrne, 2005; Koehn et al., 2005), whereby the frequencies of relative positions of the phrase pairs are extracted from the training corpus and used as additional inputs to the decoder. Adding a lexicalised reordering model has been shown to consistently improve the translation quality for several language pairs (Koehn et al., 2005). This generative learning approach is fast but may suffer from the data sparseness problem since many phrase pairs occur only once (Nguyen et al., 2009).

Figure 3.2 shows the impact of reordering models when we increase our parallel corpus (see Section 8.1). We started with a small proportion of our Arabic-English corpus and then we double it each time until the full corpus is reached. There is a nearly linear improvement which agrees with the findings of Turchi et al. (2012). Note that the gap between the baseline system (Moses) and the system without a reordering model increases as we have more data. This is because reordering models have seen more examples of phrase movements.

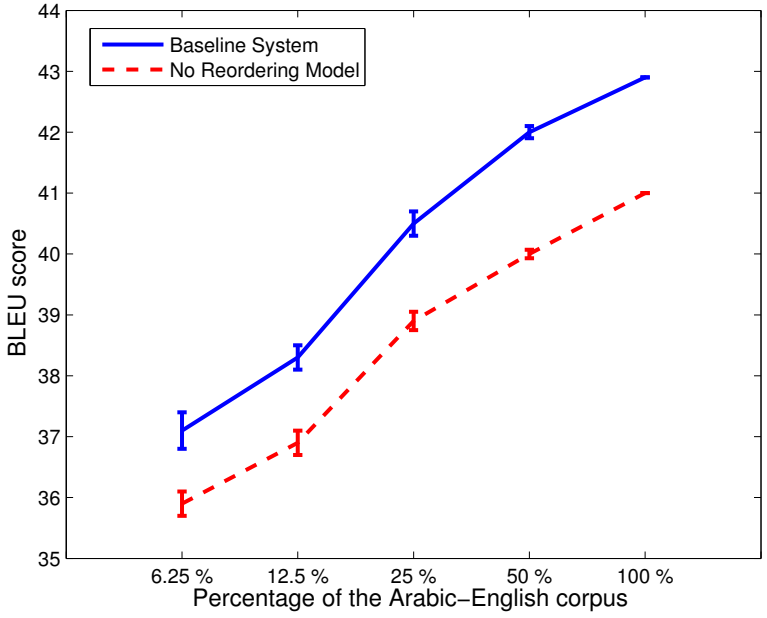

FIGURE 3.2: BLEU scores of two SMT systems (with and without a reordering model) with different sizes of our Arabic-English corpus.

Building on this, some researchers have borrowed powerful ideas from the machine learning literature, to pose the phrase movement problem as a prediction problem using contextual input features whose importance are modelled as weights of a linear classifier. This is referred to as discriminative learning approach.

In the following sections, we explain these two approaches in more detail and review the previous work.

3.1 Generative Reordering Models

Phrase reordering modelling involves formulating phrase movements as a classification problem where each phrase position considered as a class (Tillmann, 2004). Some researchers classified phrase movements into three categories (monotone, swap, and discontinuous) but the classes can be extended to any arbitrary number (Koehn and Monz, 2005). Table 3.1 shows Arabic-English phrase pairs in different orientations.

Sentence pair 1 (monotone orientation):
AR: wElm mn mSAdr mqrbp mn Alr}Asp An mbArk **Astqbl** AHmd nZyf fy mnzlh
EN: Sources close to the presidency said Mubarak **received** Nazif at his home
Sentence pair 2 (swap orientation):
AR: fy AjtmAE HA$d mE AlTlbp fy jAmEp ThrAn **Astqbl** xAtmy AstqbAlA fAtrA
EN: At a gathering with students in Tehran University, Khatami **was confronted with** a cold reception
Sentence pair 3 (discontinuous orientation):
AR: *kr An AlbAbA ywHnA bwls AlvAny **Astqbl** kAstrw llmrp AlAwlY fy rwmA fy nwfmbr
EN: The pope **met** for the first time in Rome with President Fidel Castro in November

TABLE 3.1: Three Arabic-English phrase pairs sharing the Arabic word "Astqbl" in different orientations.

A lexicalised reordering model is estimated by relative frequency where each phrase pair $(\bar{f}_i, \bar{e}_i)$ with such an orientation (o_k) is counted and then normalised to yield the probability as follows:

$$p(o_k|\bar{f}_i, \bar{e}_i) = \frac{\text{count}(\bar{f}_i, \bar{e}_i, o_k)}{\sum_o \text{count}(\bar{f}_i, \bar{e}_i, o)}. \tag{3.1}$$

The orientation of a current phrase pair is defined with respect to the previous target text. In the literature, there are three types of lexicalised reordering models (word-based, phrase-based and hierarchical) based on the definition of phrase orientation.

3.1.1 Word-based Reordering Model

Koehn (2010a) defined reordering distance as "the number of words skipped (either forward or backward) when taking foreign words out of sequence". Reordering distance is calculated as follows:

$$\text{d} = \text{start}_\text{i} - \text{end}_{\text{i}-1} - 1, \tag{3.2}$$

where (start_i) is defined as the position of first word in the source phrase that translates to the current target phrase. $(\text{end}_{\text{i}-1})$ is defined as the position of last word in the source phrase that translates to the previous target phrase. The distance is usually simplified to split segments (i.e. classes) to make up phrase orientation set O.

3.1.2 Phrase-based Reordering Model

The model presented in Tillmann (2004) is similar to the word-based orientation model presented above, except that it analyses adjacent phrases rather than specific word alignments to determine orientations.

Many researchers classified phrase movements into three categories (monotone, swap, and discontinuous) defined as follows. Monotone orientation is concluded if the previously translated source phrase occurs immediately before the current source phrase. Swap orientation is concluded if the previously translated source phrase occurs immediately after the current source phrase. Otherwise, an orientation is set to discontinuous.

3.1.3 Hierarchical Reordering Model

The lexicalised reordering model has been extended to tackle long-distance reorderings (Galley and Manning, 2008). This takes into account the hierarchical structure of the sentence when considering such an orientation. Certain examples are often used to motivate syntax-based systems were handled by this hierarchical model, and the approach is shown to improve translation performance for several translation tasks. An additional appeal of the method is the low computing cost.

Word alignments are analysed beyond adjacent phrases as shown in Figure 3.3. For example, Monotone orientation is concluded if the previously translated source phrases (i.e. no constraint on maximum phrase length) occurs immediately before the current source phrase. Swap orientation is concluded if the previously translated source phrases (i.e. no constraint on maximum phrase length) occurs immediately after the current source phrase. Otherwise, an orientation is set to discontinuous.

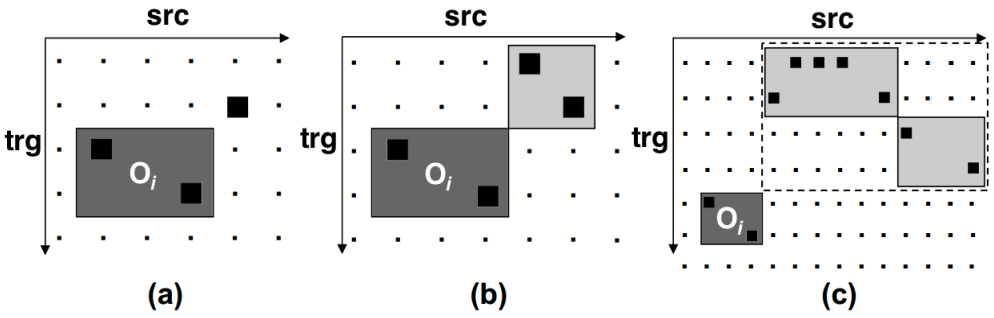

FIGURE 3.3: An example given by Galley and Manning (2008) to illustrate the differences between word-based, phrase-based, and hierarchical reordering models. Dark blocks represent current phrases and light blocks represent previously translated phrases. In (a), the orientation of the current phrase is swap according to all three models. In (b), the orientation is discontinuous according to the word-based model and swap according to the rest. In (c), the orientation is swap according to the hierarchical model and discontinuous according to the rest.

3.2 Discriminative Reordering Models

Despite the fact that the lexicalised reordering model is always biased towards the most frequent orientation for such a phrase pair, it may suffer from a data sparseness problem since many phrase pairs occur only once (Nguyen et al., 2009). In our experiments, for example, less than 15% of Arabic-English phrase pairs are repeated in the text. Moreover, the context of a phrase might affect its orientation, which is not considered as well.

Adopting the idea of predicting orientation based on content, it has been proposed to represent each phrase pair by linguistic features as reordering evidence, and then train a classifier for prediction. The maximum entropy classifier is a popular choice among many researchers as we will show below.

In general, the distribution of phrase orientation o_k is:

$$p(o_k|\bar{f}_i, \bar{e}_i) = \frac{1}{Z} h(\mathbf{w}_k^\top \boldsymbol{\phi}(\bar{f}_i, \bar{e}_i)), \tag{3.3}$$

where $h(.)$ is a pre-defined monotonic function, $\mathbf{w}$ is a weight vector, $\boldsymbol{\phi}(.)$ is a feature vector represent a phrase pair $(\bar{f}_i, \bar{e}_i)$, and Z is a normalisation term defined as:

$$Z = \sum_{k'}^{K} h(\mathbf{w}_{k'}^\top \boldsymbol{\phi}(\bar{f}_i, \bar{e}_i)). \tag{3.4}$$

3.2.1 Maximum Entropy-based Reordering Model

Zens and Ney (2006) introduced the maximum entropy classifier for phrase reordering using the Basic Travel Expression Corpus (BTEC). Three different translation tasks were carried out: Arabic-English, Chinese-English and Japanese-English. Only two orientations were considered, left or right (i.e. monotone or swap). Although the proposed model overcomes the relative frequency model in terms of classification performance, they did not draw comparison between them in terms of translation performance. The translation results reported were between their model and the distance-based reordering model. We believe that such a comparison with a lexicalised reordering model is important because the model is faster to estimate (i.e. relative frequency) and also faster to use during translation since there is no overhead computation (i.e. retrieving probabilities from a table).

Xiong et al. (2006) also proposed a maximum entropy model to predict reordering of neighbour blocks (i.e. phrase pairs) and considered straight or inverted orientations (i.e. monotone or swap). Their experiments were carried out on Chinese-English translation

tasks using FBIS corpus which is a multilanguage collection of translated news and information from media sources outside the United States by Foreign Broadcast Information Service. The reported results were only in terms of translation performance. Similar to Zens and Ney (2006), the authors compared their model with the distance-based reordering model although they did make reference to the lexicalised reordering model.

Nguyen et al. (2009) applied the maximum entropy model to learn orientations identified by the hierarchical reordering model proposed by Galley and Manning (2008). The previous work of Zens and Ney (2006) and Xiong et al. (2006) identified such an orientation without considering the hierarchical structure of previous phrases. The authors used a relatively small English-Vietnamese corpus (0.6 million words) collected from daily newspapers. The approach achieves translation improvements over the lexical hierarchical reordering model in a test set taken from the same corpus. Despite the fact that the test set is not independent from the training data, the experiments were not repeated to evaluate the uncertainties in their results.

Xiang et al. (2011) introduced a smoothed prior probability to maximum entropy model and used multiple features based on syntactic parsing. The smoothed prior is a combination of - through interpolation weight - a global distortion probability $p(o_k)$ and a local distortion probability $p(o_k|\bar{f}_n, \bar{e}_n)$ (i.e. lexicalised reordering model). The model predicts the jump distance (up to five words) from the previously translated source word to the current source word. This method does not capture the hierarchical structure of the sentence as explained by Galley and Manning (2008). The experiments were undertaken on a large-scale Chinese-English translation task (one million sentence pairs). The proposed model shows improvement over a distance-based reordering model. Like the findings of Zens and Ney (2006) and Xiong et al. (2006), there is no comparison with a lexicalised reordering model.

3.2.2 Distance Phrase Reordering Model

Ni et al. (2011) considered a variety of machine learning techniques including the maximum entropy model. They introduced a perceptron-based learning approach to modelling long distance phrase movements. Similar to Xiang et al. (2011), their model predicts the jump distance (up to five words) from the previously translated source word to the current one. Inspired by Koehn and Monz (2005), they formulate phrase movements through a relative distance. The Distance Phrase Reordering (DPR) model considers reordering relative to the previous phrase in the source sentence. The formulation was simplified to limited classes to make up phrase orientation set O as shown in Figure 3.4. In three-class orientation, $O = \{\mathrm{d} < 0, \mathrm{d} = 0, \mathrm{d} > 0\}$ and in five-class orientation $O = \{\mathrm{d} \leq -5, -5 < \mathrm{d} < 0, \mathrm{d} = 0, 0 < \mathrm{d} < 5, \mathrm{d} \geq 5\}$.

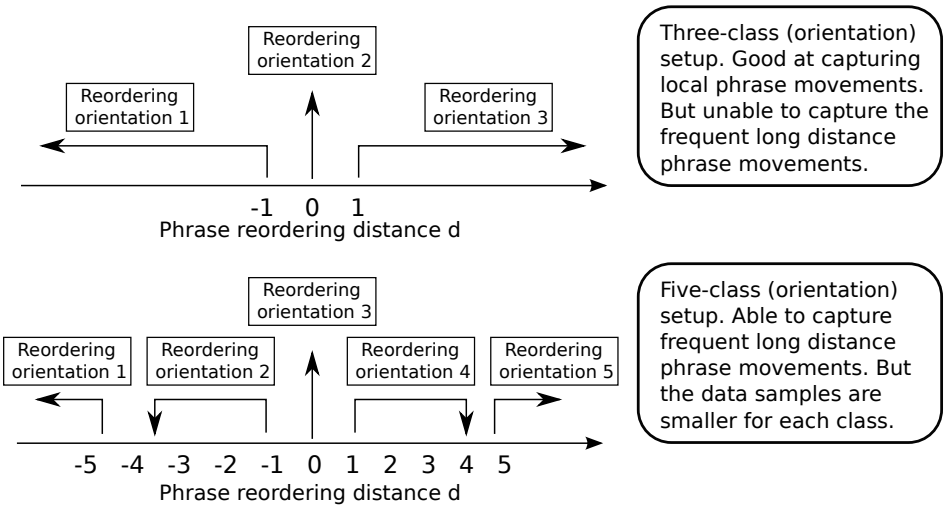

FIGURE 3.4: The phrase reordering orientations as illustrated by Ni (2010): the three-class setup (top) and the five-class setup (bottom).

Differing from the previous works, training data were divided into small independent sets where all samples share the same source phrase were considered a training set. This method breaks down the learning complexity to have as many sub-models as source phrases. Although the parameters for each sub-model are small, the overall parameters are larger than having just one model to incorporate all the data.

Several learning techniques are compared and evaluated on a Chinese-English corpus (Hong Kong laws corpus). The perceptron-based learning approach outperforms both the lexicalised reordering model and the maximum entropy model. The reported results were based on a test set taken from the same corpus. The perceptron-based learning algorithm for DPR model is described in Table 3.2. We implemented the algorithm in Matlab (see Appendix B).

Inputs: training examples $\{o_n, \boldsymbol{\phi}(\bar{f}^n, \bar{e}^n)\}_{n=1}^N$, learning rate $0 < \alpha < 1$
Initialization: t=0; $\mathbf{w}_{o,t} = \mathbf{0} \quad \forall o \in O$
Repeat
  for $i = 1, 2, \ldots, N$ **do**
    $o^* = \text{argmax}_o\{\mathbf{w}_t^T \boldsymbol{\phi}(\bar{f}^n, \bar{e}^n) + \Delta(o_n, o)\}$
    if $o^* \neq o_n$ **then**
      $\mathbf{w}_{o,t+1|o=o_n} = \mathbf{w}_{o,t|o=o_n} + \alpha\boldsymbol{\phi}(\bar{f}^n, \bar{e}^n)$
      $\mathbf{w}_{o,t+1|o=o^*} = \mathbf{w}_{o,t|o=o^*} - \alpha\boldsymbol{\phi}(\bar{f}^n, \bar{e}^n)$
      $t = t + 1$
    end if
  end for
until converge
Output: $\mathbf{w}_{t+1} \in \mathbb{R}^{dim(\boldsymbol{\phi})} \quad \forall o \in O$

TABLE 3.2: Perceptron-based learning algorithm for DPR model. Section 4.1 will explain how the training examples are generated.

In Table 3.2., the loss function $\Delta(o_n, o)$ between a pseudo orientation o and the true one defined as below:

$$\Delta(o_n, o) = \begin{cases} 0 & \text{if } o = o_n \\ 0.5 & \text{if } o \text{ and } o_n \text{ are close in } O \\ 1 & \text{otherwise} \end{cases} \tag{3.5}$$

3.2.3 Sparse Reordering Features

Previously, we learn a reordering model by estimating the weights of their parameters (i.e. reordering features) that maximise the likelihood. Here, we want to learn directly how useful these reordering features in terms of an evaluation metric (e.g. BLEU). The bottleneck of this approach is the computational complexity and the memory consumption.

Recently, Cherry (2013) proposed using sparse features to optimise BLEU with the decoder instead of training a classifier independently. The reported results shows that sparse decoder features are superior to maximum entropy classifier. Context features around each phrase pair were not considered in this work.

Sparse reordering features are proposed as a better alternative method. However, they are estimated on a small-scale because the tuning process is expensive. On the other hand, likelihood estimation of the reordering models can be trained on the whole data. Therefore, reordering models are still important. In our experiments, sparse features are used along with the reordering models in order to benefit from the two approaches.

3.3 Chapter Summary

Phrase reordering modelling involves three parts: formulating phrase movements, learning agent and phrase linguistic features. Phrase movements can be formalised as a classification problem where each phrase position considered as class. Koehn and Monz (2005) classify phrase movements into three categories (monotone, swap, discontinuous). There are three types of reordering models (word-based, phrase-based and hierarchical) based on the definition of phrase orientation.

In the learning part, Koehn and Monz (2005) propose a lexicalised reordering model estimated by relative frequency method. Zens and Ney (2006) and other use maximum entropy framework. Ni et al. (2011) apply max-margin-structure (MMS) model using perceptron-based algorithm. In chapter 5 and 6, we explore several machine learning algorithms to learn a reordering model.

In the sample representation part, a lexicalised reordering model considers the whole phrase as a sample while in discriminative reordering models phrases and their context are represented by linguistic features. They can be word sequences, part-of-speech tags (Zens and Ney, 2006) and statistical features (Tillmann and Zhang, 2005). In the next chapter, we present different ways of extracting features including a novel phrase representation we found to be effective. We also discusses several methods to reduce feature space.

Chapter 4

Feature Extraction and Selection

Many discriminative reordering models have been proposed to replace the lexicalised reordering model as shown in chapter 3. The reported results showed consistent improvement in terms of various translation metrics. These models mainly differ in two aspects: feature expression and learning agent.

In this chapter, we demonstrate how linguistic features can be expressed as reordering evidences for each phrase pair. We also discuss how to select the most informative evidences in order to reduce the feature space of our large-scale problem and maximise the ability of prediction.

4.1 Feature Extraction

The problem of phrase reordering is usually simplified by classifying phrase movements into three categories (monotone, swap, discontinuous). To extract phrase pairs along with their orientation classes, we use the `extract` tool that comes with the Moses [1] toolkit (Koehn et al., 2007). This tool requires word alignments between source and target sentences which can be produced by GIZA++ (an open source alignment tool).

Each extracted phrase pair might be represented by linguistic features as follows:

- Ni et al. (2011) use n-gram words within a window around the source phrase and the target phrase. The features are also distinguished by their positions.
- Xiong et al. (2006) define linguistic features based on the first words of the current phrase pair and the previous one, which they called the boundary of neighbour blocks. Collocation features[2] were also defined which comprise the combining of these first words.

[1] Moses is an open source toolkit for statistical machine translation (www.statmt.org/moses/).
[2] There is no position restriction as in n-gram features where words have to be consecutive.

- Nguyen et al. (2009) choose three kinds of linguistic feature taken from the current source phrase and the previous one; the headword of phrase, part of speech tag of the headword, and the syntactic label of phrase. These six features are also combined (current with previous) to have three additional features. Therefore each phrase pair has nine features in total.
- Zens and Ney (2006) represent each phrase pair as linguistic features (words and word classes) collected within a window around j and i, where j is a word position in the source phrase aligned to the last word position i in the target phrase. Note that j is not always the end of the source phrase, which means a small window like the one they have used (three positions) might not cover the context.
- Xiang et al. (2011) use multiple features based on syntactic parsing.

Table 4.1 presents a small example with three Arabic-English sentence pairs to illustrate the feature extraction process. We are interested here in the Arabic word "Astqbl" and the orientation of its translation. Each phrase pair is represented by context features within a one-word window around "Astqbl" (i.e. a word before and after).

In our experiments, each extracted phrase pair is represented by the following linguistic features as shown in Table 4.2:

- Boundaries of the phrase pair as some researchers did (Xiong et al., 2006; Nguyen et al., 2009). We found that the impact of adding features from the whole phrase pair is minimal.
- Aligned source and target words in a phrase pair. Each word alignment is a feature.
- Words within a window around the source phrase to capture the context. We did not consider the context of the target phrase as in Ni et al. (2011) for two reasons. First, we found experimentally that adding these features might increase the ability of prediction but this small improvement is not noticeable in a translation system. Second, which is more important, extracting these features during the decoding phase decreases the speed of a translation system. This is because target context is dynamic while source context is static (i.e. the source sentence is given).

As discussed in chapter 3, most researchers build one reordering model for the whole training set (Zens and Ney, 2006; Xiong et al., 2006; Nguyen et al., 2009; Xiang et al., 2011). Ni et al. (2011) simplified the learning problem to have as many sub-models as source phrases. Training data were divided into small independent sets where samples having the same source phrase are considered a training set. Source phrases that have few samples (less than 10) are discarded because they cannot be generalised very well. Table 4.3 demonstrates the two learning approaches for a toy corpus and how the feature space is constructed.

Sentence pair 1 (monotone orientation):
AR: wElm mn mSAdr mqrbp mn Alr}Asp An mbArk **Astqbl** AHmd nZyf fy mnzlh
EN: Sources close to the presidency said Mubarak **received** Nazif at his home

Sentence pair 2 (swap orientation):
AR: fy AjtmAE HA$d mE AlTlbp fy jAmEp ThrAn **Astqbl** xAtmy AstqbAlA fAtrA
EN: At a gathering with students in Tehran University, Khatami **was confronted with** a cold reception

Sentence pair 3 (discontinuous orientation):
AR: *kr An AlbAbA ywHnA bwls AlvAny **Astqbl** kAstrw llmrp AlAwlY fy rwmA fy nwfmbr
EN: The pope **met** for the first time in Rome with President Fidel Castro in November

Context features: 1. mbArk 2. AHmd 3. ThrAn 4. xAtmy 5. AlvAny 6. kAstrw

Bag-of-words expression:
Each feature is represented as binary number (1=exist, 0=not exist). The feature order is disregarded in this method. It was used by Zens and Ney (2006).

$\phi(\bar{f}_1, \bar{e}_1) = 1\ 1\ 0\ 0\ 0\ 0 \qquad \phi(\bar{f}_2, \bar{e}_2) = 0\ 0\ 1\ 1\ 0\ 0 \qquad \phi(\bar{f}_3, \bar{e}_3) = 0\ 0\ 0\ 0\ 1\ 1$

In this representation, there is $\mathbf{w}_k$ for each orientation. They can be combined in one long vector (i.e. $\mathbf{w} = [\mathbf{w}_1^\top, \mathbf{w}_2^\top, \ldots, \mathbf{w}_K^\top]^\top$) by using a joint feature vector $\phi(\bar{f}_n, \bar{e}_n, o_k)$ decomposed via the orthogonal feature representation. This representation simply means there is no crosstalk between two different feature vectors where for example $\phi(\bar{f}_n, \bar{e}_n, o_1) = [\phi(\bar{f}_n, \bar{e}_n)\, 0 \ldots 0]$. (see Rousu et al. (2006) for more details).

	mono	swap	disc.
$\phi(\bar{f}_n, \bar{e}_n, o_k)$	1 2 3 4 5 6	1 2 3 4 5 6	1 2 3 4 5 6
$\phi(\bar{f}_1, \bar{e}_1, \text{mono})$	= 1 1 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0 0
$\phi(\bar{f}_2, \bar{e}_2, \text{swap})$	= 0 0 0 0 0 0	0 0 1 1 0 0	0 0 0 0 0 0
$\phi(\bar{f}_3, \bar{e}_3, \text{disc.})$	= 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 1 1

Position-dependent expression:
Feature order is expressed by representing each feature twice (before and after). Features with hat occur after the word "Astqbl". The method was used by Ni et al. (2011)

	mono	swap	disc.
$\phi(\bar{f}_n, \bar{e}_n, o_k)$	1 $\hat{1}$ 2 $\hat{2}$ 3 $\hat{3}$ 4 $\hat{4}$ 5 $\hat{5}$ 6 $\hat{6}$	1 $\hat{1}$ 2 $\hat{2}$ 3 $\hat{3}$ 4 $\hat{4}$ 5 $\hat{5}$ 6 $\hat{6}$	1 $\hat{1}$ 2 $\hat{2}$ 3 $\hat{3}$ 4 $\hat{4}$ 5 $\hat{5}$ 6 $\hat{6}$
$\phi(\bar{f}_1, \bar{e}_1, \text{mono})$	= 1 0 0 1 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0
$\phi(\bar{f}_2, \bar{e}_2, \text{swap})$	= 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 1 0 0 1 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0
$\phi(\bar{f}_3, \bar{e}_3, \text{disc.})$	= 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 1 0 0 1

TABLE 4.1: Two feature expressions for three Arabic-English phrase pairs sharing the Arabic word "Astqbl". Arabic is written by Buckwalter transliteration (Buckwalter, 2004)

Sentence pair:

Foreign sentence **f** : $f_1\ f_2$ $_1$ $f_3\ f_4\ f_5$ $_2$ f_6 $_3$.

English sentence **e** : e_1 $_1$ $e_2\ e_3$ $_3$ $e_4\ e_5$ $_2$.

Extracted phrase pairs $(\bar{\mathbf{f}}, \bar{\mathbf{e}})$:

$\bar{f}_i$	$\bar{e}_i$	o_i	alignment	context
$f_1\ f_2$	e_1	mono	0-0 1-0	$+f_3$
$f_3\ f_4\ f_5$	$e_2\ e_3$	swap	0-1 2-0	$-f_2\ +f_6$
f_6	$e_4\ e_5$	disc.	0-0 0-1	$-f_5$

All alignment and context features:

1. f_1&e_1 2. f_2&e_1 3. $+f_3$ 4. f_3&e_5 5. f_5&e_4 6. $-f_2$ 7. $+f_6$
8. f_6&e_2 9. f_6&e_3 10. $+f_5$

Bag-of-words representation:

a phrase pair is represented as a vector where each feature is a discrete number (0=not exist).

$\phi(\bar{f}_i, \bar{e}_i)$		1 2 3 4 5 6 7 8 9 10
$\phi(\bar{f}_1, \bar{e}_1)$	=	1 1 1 0 0 0 0 0 0 0
$\phi(\bar{f}_2, \bar{e}_2)$	=	0 0 0 1 1 1 1 0 0 0
$\phi(\bar{f}_3, \bar{e}_3)$	=	0 0 0 0 0 0 0 1 1 1

TABLE 4.2: A generic example of our phrase pair extraction and representation.

Toy corpus (source ||| target):

a b c ||| A B C a b c ||| A C B a b c ||| C B A

Phrase extraction (phrase pair ||| orientation ||| features):

a | A ||| mono ||| a b A
b | B ||| mono ||| a b c A B
c | C ||| mono ||| b c B C

a | A ||| mono ||| a b A
b | B ||| swap ||| b c C B
c | C ||| disc. ||| b c A C

a | A ||| disc. ||| a b B A
b | B ||| swap ||| a b c C B
c | C ||| disc. ||| b c C

One-Model:

Training set: all extracted phrase pairs
Feature space: a b c A B C

Sub-Models:

Model a (training set):
a | A ||| mono ||| a b A
a | A ||| mono ||| a b A
a | A ||| disc. ||| a b B A
Feature space: a b A B

Model b (training set):
b | B ||| mono ||| a b c A B
b | B ||| swap ||| b c C B
b | B ||| swap ||| a b c C B
Feature space: a b c A B C

Model c (training set):
c | C ||| mono ||| b c B C
c | C ||| mono ||| b c A C
c | C ||| disc. ||| b c C
Feature space: b c A B C

TABLE 4.3: Two approaches to learn phrase reordering: one-model and sub-models.

4.2 Feature Selection

Having irrelevant or redundant features could affect the classification algorithms for three reasons (Liu and Motoda, 1998). First, the data intend to increase with more features in order to have statistical variability between classes. For example, a complete set of binary domain with five features is 32 instances while a set with ten features is 1024 instances. Therefore, the classifier needs more time to learn the larger data set. Second, irrelevant or redundant features might mislead the learning algorithms or overfit them to the data and thus have less accuracy. Third, learning algorithms with large unnecessary features tend to have a complex model which affects the accuracy and the understandability of the results.

The aim of feature selection is to find the optimal subset features which maximise the ability of prediction, which is the main concern, or simplify the learned results to be more understandable. There are many ways to measure the goodness of a feature or a subset of features; however three criteria will be discussed which are dependence, information and distance measures.

The first assumption to select useful features for prediction might be ranking the features individually according to some criteria. Because of simplicity, scalability and empirical success of feature ranking, many selection algorithms include it and also many researchers tend to use it as a baseline. Feature Ranking is also robust against overfitting as a result of its univariate method. However, two disadvantages are associated with this univariate method. First, redundant features are likely to be selected. Second, some valuable features might be lost during the process to filter out the least promising variables (Guyon and Elisseeff, 2003).

An alternative approach is to search for a good combination of subset of features. This multivariate method, referred to as a wrapper, utilises the learning machine of interest as a black box to score subsets of variables according to their predictive power. The main disadvantage of wrappers is time complexity which makes it not practical for the machine translation problems.

4.2.1 Dependence Measures

Dependence measures measure the strength of a linear relationship between two variables. When there is a strong correlation between a class and a feature, the class can be predicted by knowing a value of the feature. The Pearson correlation coefficient is a famous function that calculates the covariance of a feature x and the class y divided by the product of their standard deviation (Guyon and Elisseeff, 2003). It normalises the relation value between -1 and +1 where zero value indicates no correlation while the absolute value indicates strong relation. The positive or negative values mean direct or

inverse relationship, respectively. The function is defined as:

$$R(X,Y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} = \frac{cov(x,y)}{std(x)std(y)}, \tag{4.1}$$

where *cov* denotes the covariance and *std* the standard deviation.

The major disadvantage of correlation criteria is that they cannot detect nonlinear patterns as shown in Figure 4.1.

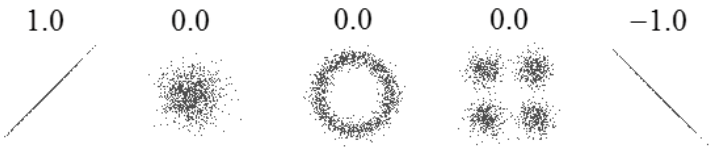

FIGURE 4.1: Values of R(x,y) for different patterns.

4.2.2 Information Measures

Information criteria are based on the concept of entropy which is the amount of randomness. The distribution of a fair coin, for example, is completely random so the uncertainty of the coin is very high. However, a biased coin is less random and thus uncertainty is lower which means higher chance to guess the next flip. Therefore, with less uncertainty the gained information is more and the following equation calculates the entropy of a variable X (MacKay, 2002):

$$H(X) = -\sum_{x} p(x) \log p(x). \tag{4.2}$$

The mutual information of a feature X can be measured by calculating the difference between the prior uncertainty of the class variable Y and the posterior uncertainty after using the feature as follows (MacKay, 2002):

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) \\ &= \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \end{aligned} \tag{4.3}$$

The advantage of mutual Information over other criteria is the ability to detect nonlinear pattern. The disadvantage is its bias towards higher arbitrary features; however this problem can be solved by normalising the information as follows (Estévez et al., 2009):

$$I_{norm}(X;Y) = \frac{I(X;Y)}{\min(H(X), H(Y))} \tag{4.4}$$

4.2.3 Distance Measures

There are many names for these criteria such as separability, divergence, or discrimination measures. Many ways have been introduces to measure the feature distance and a typical type is using the class-conditional density. For two-class problem, the distance $D(X)$ is the different between two probabilities of a feature X given the first class and the second class (Liu and Motoda, 1998). The maximum value of $D(X)$ value is one while the minimum is zero. The ability of a feature to separate the two classes increases as much as $D(X)$ increases. When $P(X|c_1)$ and $P(X|c_2)$ are similar (i.e. $D(X)$ is near to 0), the feature X has very week role in separating the two classes. Note that distance measures are similar to information measures in principle.

4.3 Chapter Summary

We covered the process of reordering feature extraction from a parallel corpus. We reviewed several methods (Zens and Ney, 2006; Xiong et al., 2006; Nguyen et al., 2009; Ni et al., 2011) and propose novel features namely word alignment features. We also give a short overview of the popular techniques for feature selection. We have chosen the normalised mutual information for our naive Bayes as we will see in Section 7.2.1. In the next two chapter, we discuss a very important aspect of phrase reordering modelling which is choosing a learning agent or a classifier. Then in Chapter 7, we will have a comprehensive comparison between these classifiers.

Chapter 5

Machine Learning for Machine Translation

Machine learning technologies have a successful history in many data-driven applications. They are powerful and can explore complex problems. However the complexity of the algorithms can limit their use. On the other hand, the simplicity of statistical methods in machine translation is one of the main reasons for their achievements since large corpora need fast methods to learn from them. Although statistical methods have some weaknesses but the amount of data they could process helps to strengthen these methods to some extent. Nonetheless, bringing machine translation close to human is beyond the ability of statistical methods. The challenge in developing machine learning methods for machine translation is adapting complex learning algorithms to process large corpora.

We have seen in Chapter 3 that phrase reordering problem can be formulated as a classification problem. In this chapter, we discuss a couple of classifiers. Before that, we give an introduction to machine learning in general and discuss the Bayesian approach to machine learning.

5.1 Machine Learning in General

Given some data, the general problem in machine learning is to learn how to relate a set of feature variables and associated target variables through a mathematical mapping function. This functional relationship between two spaces is called a model and the process to learn it is inference.

There are different types of data which means different learning problems. When the target variables are continuous values the problem is referred to as a regression problem. If it is discrete values that indicates class memberships, the problem is classification.

Both regression and classification comes under an umbrella term which is supervised learning since the target or target variables are provided.

Sometimes the data has no target variables and we would like to learn these missing variables. This situation of incomplete data requires unsupervised learning techniques. The target variables are represented as latent or hidden variables in such a model. There are two techniques clustering and projection where the former concerned about partitioning the data objects into groups. The later reduces the data dimensionality for feature selection or visualisation.

The following sections will be dedicated to classification problems in machine translation.

5.1.1 Probabilistic and Non-probabilistic Models

Probabilistic models explicitly take into account uncertainty represented by probability distribution over their variables. Such models can be divided into generative and discriminative. Examples for generative models are naive Bayes, mixture model and hidden Markov model. For discriminative models logistic regression and conditional random fields are typical examples.

Generative models learn the joint probability distribution over feature and target variables, $p(\mathbf{x}, \mathbf{y})$, which allows us to generate a data similar to ours. Then use Bayes' theorem to find the posterior target probabilities as follows:

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x})} \tag{5.1}$$

Discriminative models, on the other hand, build the posterior distribution directly, $p(\mathbf{y}|\mathbf{x})$. While these models yield superior performance, generative models are faster.

Non-probabilistic models find a discriminant function enable us to predict targets directly. Typical examples are the perceptron and support vector machine. The discriminant function is usually obtained by taking a linear function of the input features such that

$$f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) + w_0 \tag{5.2}$$

where $(\mathbf{w}, w_0)$ denotes a weight vector that maps the input features to the targets. $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ is a joint feature vector $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ decomposed via the orthogonal feature representation. This representation simply means there is no crosstalk between two different feature vectors where $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}_1)$=[$\boldsymbol{\phi}(\mathbf{x})$ 0 ... 0] and $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}_2)$=[0 ... 0 $\boldsymbol{\phi}(\mathbf{x})$], .. etc (see Rousu et al. (2006) for more details).

5.2 Bayesian Inference in Probabilistic Models

The uncertainty in probabilistic models' decisions goes further in Bayesian inference towards the models themselves. Given a model with such parameters (θ), we would like to quantify how probable or likely is our data ($\mathcal{D}$) to be generated from this model. The quantity is called likelihood and maximising it leads to estimate good parameters for the model. This estimation method is known as maximum likelihood.

The uncertainty in our model is expressed by putting a prior over the parameters. Then we marginalise the likelihood and the prior over parameter space to have the probability of our data or the model evidence. It is computed as follows:

$$p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)\,\mathrm{d}\theta \tag{5.3}$$

In Bayesian setting, predicting the target probability of a new object is based on the whole θ space not just a single value of θ. This requires a posterior probability distribution over θ. By means of Bayes' theorem the posterior is:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \tag{5.4}$$

Then we integrate out to get the target probability as follows:

$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{x},\theta)p(\theta|\mathcal{D})\,\mathrm{d}\theta \tag{5.5}$$

These integrations might be computationally expensive. However, choosing the right prior for the likelihood sometimes allows us to compute the marginalisation analytically. From a mathematical point of view, the posterior can be computed analytically if we know the density is the same form as the prior. It is the case because omitting the denominator in Bayes' rule gives:

$$p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta) \tag{5.6}$$

Therefore all what we needs is to rearrange the nominator to look like the prior form. Once we know the posterior's parameters the normalising constant is known.

Some likelihood-prior pairs form the same density as the prior known as conjugate priors. Table 5.1 shows four common pairs. Next section will discuss the case when the likelihood-prior pairs are non-conjugate which results in a posterior with unknown form.

Prior	Likelihood
Gaussian	Gaussian
Beta	Bunomial
Gamma	Gaussian
Dirichlet	Multinomial

TABLE 5.1: Conjugate priors

5.2.1 Non-conjugate Models

In standard probability distributions, the normalising constant that insures the probabilities sum to one is known. When the likelihood-prior pair is non-conjugate, the posterior is not a standard distribution. This requires computing the normalising constant in order to have a probability distribution. In many cases, the normalising constant (i.e. integration) cannot be solved analytically. Therefore we have to restore to estimation techniques.

There are three options to estimate the normalising constant:

1. *MAP solution*: In this technique, instead of computing a probability distribution over the parameters, we choose the most probable θ. This maximum a posteriori estimation is not considered a very Bayesian way since the solution is based on a single point. However, it can yield good results when the distribution is peaked around small area without multiple modes. Since the nominator in Bayes' rule (i.e. likelihood and prior) is proportional to the posterior, its maximum correspond to the posterior's maximum.

2. *Sampling Methods*: They are stochastic techniques that sample directly from the posterior. Given infinite computational resources, exact distribution can be achieved. However, the computational cost limits their use to small-scale problems. A popular class of sampling methods is called Markov chain Monte Carlo. They walk randomly through the distribution space and generate a Markov chain of samples. Metropolis–Hastings and Gibbs sampling algorithms are common random walk MCMC methods.

3. *Approximate Inference*: It is alternative to sampling methods based on a deterministic approximation to the posterior with other density can be solved analytically. This advantage makes the technique applicable for large-scale problems. However, the technique can never generate exact posterior therefore choosing a density close enough to the true one is critical. There are two types of approximations based on local or global criteria. Laplace approximation, for example, is a local Gaussian approximation. Differently, variational Bayes use global criteria which has more recent interest and also its alternative known as expectation propagation.

5.3 Multiclass Perceptron

A perceptron is a non-probabilistic binary classifier based on a discriminant function of the input features $\mathbf{x}_i$ such that

$$h(\mathbf{x}_i) = g(\mathbf{w}^\top \mathbf{x}_i), \tag{5.7}$$

where g is a binary activation function and $\mathbf{w}$ denotes a weight vector that maps the input features to the targets. For multiclass problems, one can use either one-versus-rest or one-versus-one strategies. Alternatively, we directly train a multiclass classifier as proposed by Crammer and Singer (2001). The discriminant function is defined as follows:

$$h(\mathbf{x}_i) = \arg\max_{y_k} \left\{ \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i, y_k) \right\}, \tag{5.8}$$

where $\boldsymbol{\phi}(.,.)$ is a joint feature vector (see Section 4.1).

Ni et al. (2011) proposed a multiclass perceptron with a distance function $\Delta(y_i, y_k)$ between classes for the phrase reordering problem. They aruge that misclassified samples should not have the same loss as some orientations are closer to others than the rest as shown in Figure 5.1

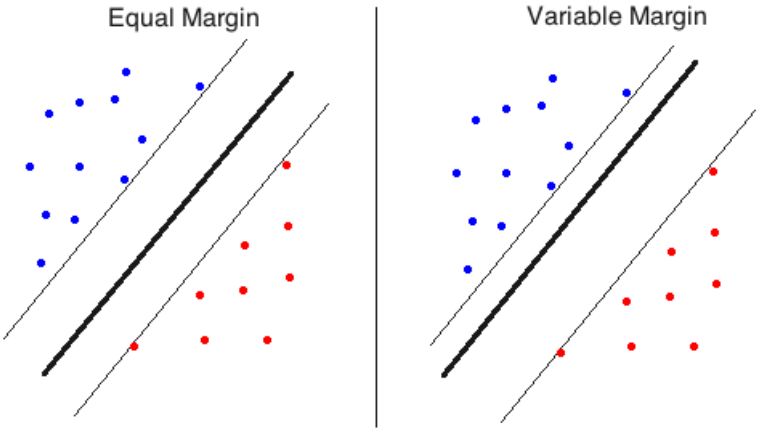

FIGURE 5.1: Illustration of an equal and variable margin hyperplane in two-class problem.

The approach tries to solve the following optimisation problem:

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \ell(\mathbf{x}_i, y_i, \mathbf{w}), \tag{5.9}$$

where $\ell(.)$ is a hinge loss function with variable margin defined as:

$$\ell(\mathbf{x}_i, y_i, \mathbf{w}) = \max\left(0\ ,\ \max_{y_k \neq y_i}\left\{\Delta(y_i, y_k) + \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i, y_k)\right\} - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i, y_i)\right)\ . \tag{5.10}$$

Algorithm 1 shows how $\mathbf{w}$ is estimated by online learning method. The advantage of perceptron is that the complexity is linear in N which makes it applicable for large scale problems such as machine translation. Although the algorithm is faster than many learning methods, it might tend to overfit the training data, and an *early stopping strategy* is one of the solutions adopted to address this issue (Ni et al., 2011).

Algorithm 1 Online learning for multiclass perceptron with variable margin.

Require: training set $S = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ and the learning rate $0 < \alpha < 1$

Initialisation: $\mathbf{w} \leftarrow \mathbf{0}$
repeat
    Randomly shuffle S
    for $i = 1$ **to** N **do**
        $y* \leftarrow \text{argmax}_{y_k}\{\Delta(y_i, y_k) + \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i, y_k)\}$
        if $y^* \neq y_i$ **then**
            $\mathbf{w} \leftarrow \mathbf{w} + \alpha\,(\,\boldsymbol{\phi}(\mathbf{x}_i, y_i) - \boldsymbol{\phi}(\mathbf{x}_i, y^*)\,)$
        end if
    end for
until converge
Output: $\mathbf{w} \in \mathbb{R}^{dim(\boldsymbol{\phi})}$

5.4 Multinomial Logistic Regression

Multinomial logistic regression, also known as softmax regression or maximum entropy, is a probabilistic model for multiclass problem. It is a popular classifier in the community of machine translation. The class probability is given by:

$$p(y_k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{k'} \exp(\mathbf{w}_{k'}^\top \mathbf{x})} \tag{5.11}$$

Unlike naive Bayes, the posterior class distribution is modelled directly without the need to model the joint distribution. Hence it is a discriminative model.

The model's parameters are estimated by maximum likelihood. Since each class has its own parameters, the likelihood have to be maximised according to all of them. To do that, we write the function using the 1-of-C coding scheme in which $\mathbf{t}_i$ is a zero vector except y_{ik} equals one which indicates an object is belonging to that class (Bishop, 2006). Then the likelihood is expressed as:

$$p(\mathbf{y}|\mathbf{X}) = \prod_{i=1}^{N}\prod_{k}^{K} p(y_k|\mathbf{x}_i)^{t_{ik}} \tag{5.12}$$

Analogous to naive Bayes, we put a prior over $\mathbf{w}_k$ to have a Bayesian model. Unfortunately, there is no conjugate prior for our likelihood which means the posterior is not a standard distribution. As discussed in section 5.2.1, there are three ways to overcome that. Since the computational cost in large-scale problems like machine translation is important, we are going to present here maximum a posterior (MAP) estimation.

Omitting the denominator in Bayes' rule makes the posterior proportional to the prior times the likelihood as follows:

$$p(\mathbf{w}_1, ..., \mathbf{w}_K|\mathbf{X}, \mathbf{y}) \propto p(\mathbf{w}_1, ..., \mathbf{w}_K)p(\mathbf{y}|\mathbf{X}, \mathbf{w}_1, ..., \mathbf{w}_K) \tag{5.13}$$

MAP solution is based on the most probable parameters in the posterior distribution. Therefore maximising the right side yields the solution. We choose a Gaussian density $N(\mathbf{0}, \sigma^2\mathbf{I})$ for the prior. The covariance is motivated by analytical convenience. It makes the parameters independent from each other and shared the same hyperparameter (i.e. σ^2). Now, taking the partial derivative gives (Rogers and Girolami, 2011):

$$\frac{\partial \log[p(\mathbf{w}_1, ..., \mathbf{w}_K)p(\mathbf{y}|\mathbf{X}, \mathbf{w}_1, ..., \mathbf{w}_K)]}{\partial \mathbf{w}_k} = -\frac{1}{\sigma^2}\mathbf{w}_k + \sum_{i=1}^{N}[t_{ik} - p(y_k|\mathbf{x}_i)]\mathbf{x}_i \tag{5.14}$$

The solution is not closed-form so we use an optimisation algorithm such as gradient ascent to obtain $\mathbf{w}_k$. The Newton-Raphson algorithm is another option which requires the second derivatives (Hessian matrix). It finds the solution faster when computing Hessian is not expensive. Note that the partial derivative in maximum likelihood solution is similar to MAP solution but without $(-\frac{1}{\sigma^2}\mathbf{w}_k)$ which serves as regularisation term. It prevents some parameters growing very large leading to overfitting.

Table 2 presents how to learn the model by gradient ascent. It is also implemented in Matlab (see Appendix B). The algorithm without the regularisation term is same as perceptron if we assign hard decision probability, zero or one.

In our experiments, we used a more advanced optimisation algorithm proposed by Andrew and Gao (2007)[1]. Their algorithm optimises both L_1-regularised and L_2-regularised log-likelihood based on L-BFGS algorithm. The L_1 regularisation is equivalent to adding Laplacian prior over the model's parameters.

[1]We have used the authors' implementation of L-BFGS algorithm which is available at http://homes.cs.washington.edu/∼galen/

Algorithm 2 Gradient ascent learning for Multinomial Logistic Regression.

Require: training set $S = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$, σ and the learning rate $0 < \alpha < 1$

Initialisation: $\mathbf{w}_k \leftarrow \mathbf{0} \quad \forall k$
repeat
  $\mathbf{q}_k = -\frac{1}{\sigma^2}\mathbf{w}_k \quad \forall k$
  for $i = 1$ **to** N **do**
    for $k = 1$ **to** K **do**
      $\mathbf{g}_k \leftarrow \mathbf{g}_k + [t_{ik} - p(y_k|\mathbf{x}_i)]\mathbf{x}_i$
    end for
  end for
  $\mathbf{w}_k \leftarrow \mathbf{w}_k + \alpha\mathbf{g}_k \quad \forall k$
until converge
Output: $\mathbf{w}_k \in \mathbb{R}^{dim(\mathbf{x})}$

5.4.1 Relation to the Maximum Entropy Principle

In natural language processing literature, the maximum entropy method is very popular. It is used as a means of estimating probability distributions from data. On the other hand, multinomial logistic regression estimated by maximum likelihood is well known to yield the same distribution (MacKay, 2002).

The principle of maximum entropy states that the best distribution of a data is the one assumes nothing about what we do not know (i.e. uniform distribution). Since entropy estimates the uncertainty, maximising entropy of a distribution subject to constraints (i.e. data) keeps the uniformity for the unknown data. The optimisation problem is given by:

$$\begin{aligned} &\max \quad H(p) \\ s.t. \quad &< f(\mathbf{x}, \mathbf{y}) >_p = < f(\mathbf{x}, \mathbf{y}) >_{data} \end{aligned} \tag{5.15}$$

where $< \, . \, >_p$ is the expectation with respect to distribution p(.) and $< \, . \, >_{data}$ is the expectation with respect to the empirical data (i.e. the average). Converting the problem to unconstrained by introducing Lagrange multipliers, the distribution has the form:

$$p(y_k|\mathbf{x}_n)_{\text{Maxent}} = \frac{1}{Z}\exp(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n, y_k)) \tag{5.16}$$

It is in fact the same as multinomial logistic regression. Note that the parameters $\mathbf{w}$ are the same for each class because a joint feature vector $\boldsymbol{\phi}(\mathbf{x}_n, y_k)$ is used. This results to a long vector composed of all classes' parameters (i.e. $\mathbf{w} = [\mathbf{w}_1^\top ... \mathbf{w}_K^\top]^\top$).

When the parameters estimated by maximum likelihood, they satisfy the same constraints in maximum entropy principle. This means both learning methods are going to yield identical results. We show that by taking partial derivative of the log-likelihood as follows:

$$\begin{aligned}
\frac{\partial \log p(\mathbf{y}|\mathbf{X})}{\partial \mathbf{w}} &= \sum_{n}^{N}\sum_{k}^{K}[t_{nk} - p(y_k|\mathbf{x}_n)]\boldsymbol{\phi}(\mathbf{x}_n, y_k) \\
&= \sum_{n}^{N}\sum_{k}^{K} t_{nk}\boldsymbol{\phi}(\mathbf{x}_n, y_k) - \sum_{n}^{N}\sum_{k}^{K} p(y_k|\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n, y_k) \\
\sum_{n}^{N}\sum_{k}^{K} p(y_k|\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n, y_k) &= \sum_{n}^{N}\boldsymbol{\phi}(\mathbf{x}_n, \mathbf{y}_n) \\
< f(\mathbf{x}, \mathbf{y}) >_p &= < f(\mathbf{x}, \mathbf{y}) >_{data}
\end{aligned} \tag{5.17}$$

5.5 Ordinal Regression

Ni (2010) suggests to extend the formulation of phrase reordering from classification problem to ordinal regression. The argument is that phrases exist in order and such a model should respect this structure. Ordinal regression is considered as a problem between classification and regression. The model predicts an ordinal variable exist on a scale from 1 to K. Unlike classification, outputs that are closer to the true class have more probability than far ones. Hence, misclassified data points are not equal.

The model is a cumulative distribution function defined as (McCullagh, 1980):

$$\begin{aligned}
0 \leq p(y_1|\mathbf{x}) \leq \cdots \leq p(y_{k-1}|\mathbf{x}) \leq 1 \\
p(k-1 < y \leq k|\mathbf{x}) = p(y_k|\mathbf{x}) - p(y_{k-1}|\mathbf{x})
\end{aligned} \tag{5.18}$$

where the distribution function is related to a linear predictor by a link function (5.20) as follows:

$$\begin{aligned}
\text{link}(\, p(y_k|\mathbf{x})\,) = \theta_k - \mathbf{w}^\top\mathbf{x} \\
-\infty < \theta_1 < \cdots < \theta_{K-1} < \infty
\end{aligned} \tag{5.19}$$

Figure 5.2 illustrates how Ordinal Regression's hyperplanes are constructed for a three-class problem.

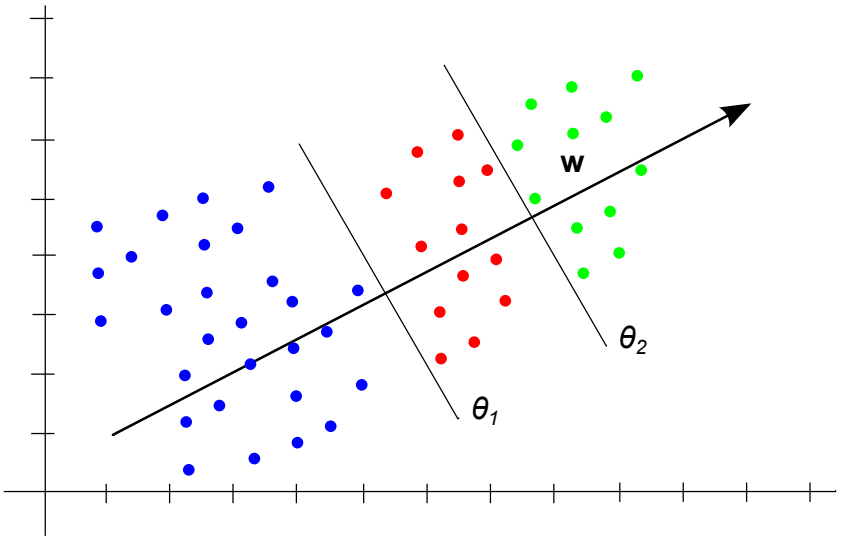

FIGURE 5.2: Ordinal Regression's hyperplanes for a three-class problem.

Below is three commonly used link functions. Their differences are shown in Figure 5.3.

$$x = \text{link}(p) = \begin{cases} \log(p/(1-p)) & \text{Logit link} \\ \log(-\log(1-p)) & \text{Complementary log-log link} \\ -\log(-\log(p)) & \text{Negative log-log link} \end{cases} \tag{5.20}$$

$$p = \text{link}^{-1}(x) = \begin{cases} \exp(x)/(1+\exp(x)) & \text{Logit link} \\ 1-\exp(-\exp(x)) & \text{Complementary log-log link} \\ \exp(-\exp(-x)) & \text{Negative log-log link} \end{cases} \tag{5.21}$$

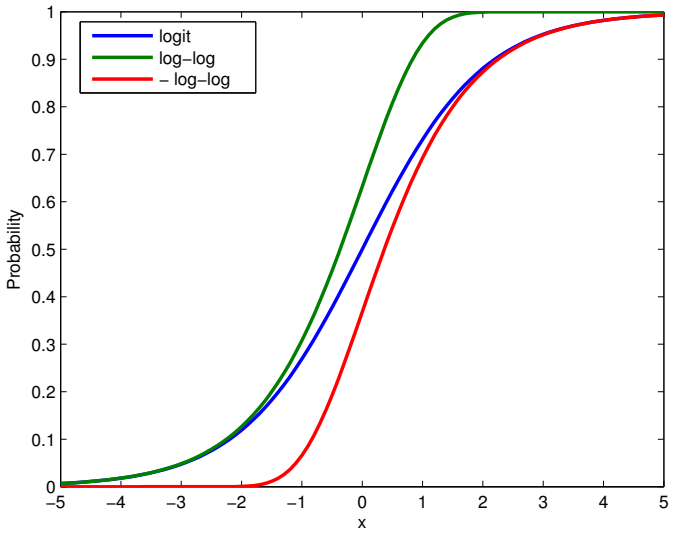

FIGURE 5.3: Three commonly used link functions.

5.5.1 Parameter Estimation

We choose logit link to define ordinal regression as follows:

$$p(y_k|\mathbf{x}_i) = \gamma_{i,k} = \frac{\exp(\theta_k - \mathbf{w}^\top \mathbf{x}_i)}{1 + \exp(\theta_k - \mathbf{w}^\top \mathbf{x}_i)} \tag{5.22}$$

The model's parameters are estimated by maximising the likelihood expressed as:

$$L(\theta, \mathbf{w}) = \prod_{i=1}^{N} \prod_{k=1}^{K} (\gamma_{i,k} - \gamma_{i,k-1})^{\delta y_i, k} \tag{5.23}$$

Taking the partial derivative with respect to each parameter gives:

$$\frac{\partial \log L}{\partial \theta_k} = \sum_{i=1}^{N} \gamma_{i,k}(1 - \gamma_{i,k})(\frac{\delta y_i, k}{\gamma_{i,k} - \gamma_{i,k-1}} - \frac{\delta y_i, k+1}{\gamma_{i,k+1} - \gamma_{i,k}}) \tag{5.24}$$

$$\frac{\partial \log L}{\partial \mathbf{w}} = \sum_{i=1}^{N} \sum_{k=1}^{K} -\mathbf{x}_i(\gamma_{i,k} - \gamma_{i,k}^2 - \gamma_{i,k-1} + \gamma_{i,k-1}^2) \frac{\delta y_i, k}{\gamma_{i,k} - \gamma_{i,k-1}} \tag{5.25}$$

Since there is no closed-form solution, an optimisation algorithm such as gradient ascent can be used as follows:

$$\theta_k^{'} = \theta_k + \alpha \frac{\partial \log L}{\partial \theta_k} \tag{5.26}$$

$$\mathbf{w}^{'} = \mathbf{w} + \alpha \frac{\partial \log L}{\partial \mathbf{w}} \tag{5.27}$$

5.6 Voted Spheres

Differing from the previous models, Voted Spheres is a non-parametric classifier proposed by Farran and Saunders (2009). It was proposed as a large-scale classifier that can produce non-linear decision function.

Voted Spheres is similar to the idea of k-nearest neighbour but a new input is classified by a majority vote of the closest spheres instead of its neighbours. These spheres are constructed from the training data points and act as a compression set. The training phase is presented in Algorithm 3 and the testing phase is presented in Algorithm 4.

Algorithm 3 Voted Spheres: training phase

Require: training set $S = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ and $R_k \in \mathbb{R}$ is a maximum radius for each class

Create an empty vector V to store spheres
for i=1 **to** N **do**
    inSphere $\leftarrow$ FALSE
    for s=1 **to** $size(V)$ **do**
        if $y_i = V_s.class$ and $\|\mathbf{x}_i - V_s.centre\| < R_{y_i}$ **then**
            inSphere $\leftarrow$ TRUE
            $V_s.weight \leftarrow V_s.weight + 1$
        end if
    end for
    if inSphere=FALSE **then**
        Add $\mathbf{x}_i$ to V as a centre of a new sphere
    end if
end for

Algorithm 4 Voted Spheres: testing phase

Require: voted spheres V

Initialise $T_k \leftarrow 0 \quad \forall k$
inSphere $\leftarrow$ FALSE
for s=1 **to** $size(V)$ **do**
    if $\|\mathbf{x} - V_s.centre\| < V_s.R$ **then**
        inSphere $\leftarrow$ TRUE
        $T_{V_s.class} \leftarrow T_{V_s.class} + V_s.weight$
    end if
end for
if inSphere=FALSE **then**
    $\hat{s} \leftarrow \text{argmax}_s \|\mathbf{x} - V_s.centre\|$
    $y \leftarrow V_{\hat{s}}.class$
else
    $y \leftarrow \text{argmax}_k \ T_k$
end if

5.7 Chapter Summary

We gave a short introduction to machine learning in general and covered a variety of classifiers. We also presented ordinal regression as an alternative method to variable margin perceptron. Finally, we have non-linear classifier can process large data. In the next chapter, we propose several large-scale classifiers which are our contributions in this thesis. In Chapter 7, we evaluate all these classification models.

Chapter 6

Large-scale Classification Models

In this chapter, we discuss our main contributions. First, we explore a generative learning approach to phrase reordering namely naive Bayes. Then, we explore recent advancements in solving large-scale classification problems in order to produce significant saving in computation and memory while preserving the accuracy. There are four classifiers will be presented: Multinomial Naive Bayes, Dual Multinomial Logistic Regression, Memory-efficient SVM and Multiclass SVM.

6.1 Multinomial Naive Bayes

Naive Bayes method has been a popular classification model of choice in many natural language processing problems (e.g. text classification). Naive Bayes is a simple classifier that ignores correlation between features, but has the appeal of computational simplicity. It is a generative probabilistic model based on Bayes' theorem as below:

$$p(y_k|\mathbf{x}_n) = \frac{p(\mathbf{x}_n|y_k)p(y_k)}{\sum_{k'} p(\mathbf{x}_n|y_{k'})p(y_{k'})}. \tag{6.1}$$

The class prior can be estimated easily as a relative frequency (i.e. $p(o_k) = \frac{N_k}{N}$). The likelihood distribution $p(\mathbf{x}_n|y_k)$ is defined based on the type of data. The classifier will be naive if we assume that feature variables are conditionally independent. The naive assumption simplifies our distribution and hence reduces the parameters that have to be estimated. In text processing, multinomial is used as a class-conditional distribution (Rogers and Girolami, 2011). The distribution is defined as:

$$p(\mathbf{x}_n|\mathbf{q}) = C \prod_m q_m^{x_{nm}} \tag{6.2}$$

where C is a multinomial coefficient,

$$C = \frac{(\sum_m x_{nm})!}{\prod_m x_{nm}!}, \tag{6.3}$$

and $\mathbf{q}$ are a set of parameters, each of which is a probability ($\sum_m q_m = 1$). Estimating these parameters for each class by maximum likelihood,

$$\underset{\mathbf{q}_k}{\operatorname{argmax}} \prod_n^{N_k} p(\mathbf{x}_n|\mathbf{q}_k), \tag{6.4}$$

will result in (Rogers and Girolami, 2011):

$$q_{km} = \frac{\sum_n^{N_k} x_{nm}}{\sum_{m'}^{M} \sum_n^{N_k} x_{nm'}}. \tag{6.5}$$

MAP estimate It is clear that q_{km} might be zero which means the probability of a new data point with nonzero feature x_{nm} is always zero because of the product in (6.3). Putting a prior over $\mathbf{q}$ is one smoothing technique. A conjugate prior for the multinomial likelihood is the Dirichlet distribution as shown in Table 5.1. It is defined as:

$$p(\mathbf{q}|\boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_m \alpha_m\right)}{\prod_m \Gamma(\alpha_m)} \prod_m q_m^{\alpha_m - 1} \tag{6.6}$$

where $\boldsymbol{\alpha}$ are a set of hyperparameters, each of which is more than zero.

The MAP estimate for q_{km} is (Rogers and Girolami, 2011):

$$q_{km} = \frac{\alpha - 1 + \sum_n^{N_k} x_{nm}}{M(\alpha - 1) + \sum_{m'}^{M} \sum_n^{N_k} x_{nm'}} \tag{6.7}$$

where M is the feature vector's length or the feature dictionary size and α is a Dirichlet parameter with a value greater than one. The derivation is in Appendix A.

Bayesian inference Instead of using a point estimate of $\mathbf{q}$ as shown previously in equation (6.7), Bayesian inference is based on the whole parameter space in order to incorporate uncertainty into our multinomial model as discussed in section 5.2. This requires a posterior probability distribution over $\mathbf{q}$ as follows:

$$\begin{aligned} p(\mathbf{x}_n|y_k) &= \int p(\mathbf{x}_n|\mathbf{q}_k) p(\mathbf{q}_k|\boldsymbol{\alpha}_k)\, \mathrm{d}\mathbf{q}_k \\ &= C \frac{\Gamma\left(\sum_m \alpha_{km}\right)}{\prod_m \Gamma(\alpha_{km})} \frac{\prod_m \Gamma(\alpha_{km} + x_{nm})}{\Gamma\left(\sum_m \alpha_{km} + x_{nm}\right)}. \end{aligned} \tag{6.8}$$

Here $\boldsymbol{\alpha}_k$ are new hyperparameters of the posterior derived by means of Bayes' theorem as follows:

$$p(\mathbf{q}_k|\boldsymbol{\alpha}_k) = \frac{p(\mathbf{q}_k|\boldsymbol{\alpha}) \prod_n^{N_k} p(\bar{f}_n, \bar{e}_n|\mathbf{q}_k)}{\int p(\mathbf{q}_k|\boldsymbol{\alpha}) \prod_n^{N_k} p(\bar{f}_n, \bar{e}_n|\mathbf{q}_k) \mathrm{d}\mathbf{q}_k}. \tag{6.9}$$

The solution of (6.10) will result in:

$$\boldsymbol{\alpha}_k = \boldsymbol{\alpha} + \sum_n^{N_k} \mathbf{x}_n. \tag{6.10}$$

For completeness we give a summary of derivations of equations (6.8) and (6.10) in Appendix A, more detailed discussions can be found in Barber (2012).

Note that the parameters are integrated out which means we no longer need to estimate $\mathbf{q}_k$ to compute the probability. This is a very important advantage in the full Bayesian approach. The training should be fast since there is no iterative learning. On the other hand, computing (6.8) during prediction might be computationally expensive. However, we factor out common terms between the numerator and the denominator in order to compute the probability efficiently as follows:

$$p(\mathbf{x}|y_k) = C \frac{\prod_m F(x_m, \alpha_{km})}{F(\sum_m x_m, \sum_m \alpha_{km})}, \tag{6.11}$$

where

$$F(a, b) = \begin{cases} 1 & \text{if } a = 0; \\ b & \text{if } a = 1; \\ b \times \cdots \times (b + a - 1) & \textit{otherwise}. \end{cases} \tag{6.12}$$

Note that the product in $F(a, b)$ is not computed across all m dimensions since the feature vector is sparse in phrase reordering problem (i.e. most of $x_m = 0$). We implemented the classifier in Matlab (see Appendix B).

6.2 Dual Multinomial Logistic Regression

Lebanon and Lafferty (2002) derived an equivalent dual problem to (6.21). Introducing Lagrange multipliers $\boldsymbol{\alpha}$, the dual becomes

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2\sigma^2}\sum_{k=1}^{K}\|\mathbf{w}_k(\boldsymbol{\alpha})\|^2 + \sum_{i=1}^{N}\sum_{k=1}^{K}\alpha_{ik}\log\alpha_{ik}, \\ \text{s.t.} \quad & \sum_{k=1}^{K}\alpha_{ik} = 1 \;\text{ and }\; \alpha_{\mathrm{ik}} \geq 0\,, \forall \mathrm{i}, \mathrm{k}, \end{aligned} \tag{6.13}$$

where

$$\mathbf{w}_k(\boldsymbol{\alpha}) = \sigma^2\sum_{i=1}^{N}(\tilde{p}_{ik} - \alpha_{ik})\mathbf{x}_i \tag{6.14}$$

As mentioned in the introduction, Yu et al. (2011) proposed a two-level dual coordinate descent method to minimize $\mathcal{D}(\boldsymbol{\alpha})$ in (6.13) but it has some numerical difficulties. Collins et al. (2008) proposed simple exponentiated gradient (EG) algorithm for Conditional Random Feild (CRF). The algorithm is applicable to our problem, a special case of CRF. The rule update is:

$$\alpha'_{ik} = \frac{\alpha_{ik}\exp(-\eta_i\,\nabla_{ik}\mathcal{D}(\boldsymbol{\alpha}))}{\sum_{k'}\alpha_{ik'}\exp(-\eta_i\,\nabla_{ik'}\mathcal{D}(\boldsymbol{\alpha}))} \tag{6.15}$$

where

$$\nabla_{ik}\mathcal{D}(\boldsymbol{\alpha}) \equiv \frac{\partial\mathcal{D}(\boldsymbol{\alpha})}{\partial\alpha_{ik}} = 1 + \log\alpha_{ik} + \left(\mathbf{w}_y(\boldsymbol{\alpha})^\top\mathbf{x}_i - \mathbf{w}_k(\boldsymbol{\alpha})^\top\mathbf{x}_i\right). \tag{6.16}$$

Here y represents the true class (i.e. $o_y = o_i$). To improve the convergence, η_i is adaptively adjusted for each example. If the objective function (6.13) did not decrease, η_i is halved for a number of trials (Collins et al., 2008). Calculating the function difference below is the main cost in EG algorithm,

$$\begin{aligned} \mathcal{D}(\boldsymbol{\alpha}') - \mathcal{D}(\boldsymbol{\alpha}) = & \sum_{k=1}^{K}\left(\alpha'_{ik}\log\alpha'_{ik} - \alpha_{ik}\log\alpha_{ik}\right) \\ & - \sum_{k=1}^{K}(\alpha'_{ik} - \alpha_{ik})\mathbf{w}_k(\boldsymbol{\alpha})^\top\mathbf{x}_i \\ & + \frac{\sigma^2}{2}\|\mathbf{x}_i\|^2\sum_{k=1}^{K}(\alpha'_{ik} - \alpha_{ik})^2. \end{aligned} \tag{6.17}$$

Clearly, the cost is affordable because $\mathbf{w}_k(\boldsymbol{\alpha})$ is maintained throughout the algorithm as follows:

$$\mathbf{w}_k(\boldsymbol{\alpha}') = \mathbf{w}_k(\boldsymbol{\alpha}) - \sigma^2(\alpha'_{ik} - \alpha_{ik})\mathbf{x}_i \tag{6.18}$$

Following Yu et al. (2011), we initialise α_{ik} as follows:

$$\alpha_{ik} = \begin{cases} (1-\epsilon) & \text{if } y_k = y_i; \\ \frac{\epsilon}{K-1} & \text{otherwise.} \end{cases} \tag{6.19}$$

where ϵ is a small positive value. This is because the objective function (6.13) is not well defined at $\alpha_{ik} = 0$ due to the logarithm appearance.

Finally, the optimal dual variables are achieved when the following condition is satisfied for all examples (Yu et al., 2011):

$$\max_k \nabla_{ik}\mathcal{D}(\boldsymbol{\alpha}) = \min_k \nabla_{ik}\mathcal{D}(\boldsymbol{\alpha}) \tag{6.20}$$

This condition is the key to accelerate EG algorithm. Unlike the primal problem (6.21), the dual variables α_{ik} are associated with each example (i.e. phrase pair) therefore a training example can be disregarded once its optimal dual variables are obtained. More data shrinking can be achieved by tolerating a small difference between the two values in (6.20). Algorithm 5 presents the overall procedure (shrinking step is from line 6-9).

Algorithm 5 Shrinking stochastic exponentiated gradient method for training dual MLR

Require: training set $S = \{\mathbf{x}_i, y_i\}_{i=1}^N$

1: Given $\boldsymbol{\alpha}$ and the corresponding $\mathbf{w}(\boldsymbol{\alpha})$
2: **repeat**
3: Randomly pick i from S
4: Claculate $\nabla_{ik}\mathcal{D}(\boldsymbol{\alpha})$ $\forall k$ by (6.16)
5: $v_i \leftarrow \max_k \nabla_{ik}\mathcal{D}(\boldsymbol{\alpha}) - \min_k \nabla_{ik}\mathcal{D}(\boldsymbol{\alpha})$
6: **if** $v_i \leq \epsilon$ **then**
7: Remove i from S
8: Continue from line 3
9: **end if**
10: $\eta \leftarrow 0.5$
11: **for** $t = 1$ **to** maxTrial **do**
12: Calculate α'_{ik} $\forall k$ by (6.15)
13: **if** $\mathcal{D}(\boldsymbol{\alpha}') - \mathcal{D}(\boldsymbol{\alpha}) \leq 0$ **then**
14: Update $\boldsymbol{\alpha}$ and $\mathbf{w}(\boldsymbol{\alpha})$ by (6.18)
15: Break
16: **end if**
17: $\eta \leftarrow 0.5\,\eta$
18: **end for**
19: **until** $v_i \leq \epsilon \quad \forall i$

6.3 Memory-efficient Support Vector Machine

Support Vector Machines (SVM) were invented by Boser et al. (1992). They form a maximum margin classifier to learn a mapping between an input and an output space. The absence of local minima and the use of kernels are key advantages in the approach. The obtained solution is sparse and dimensionally independent.

Given a training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$ where $\mathbf{x}_i \in R^n$ and $y_i \in \{-1, +1\}$, the maximum margin hyperplane $(\mathbf{w}, b)$ requires solving the following optimisation problem:

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimise}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i, \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i\,, \forall i, \\ & \xi_i \geq 0\,, \forall i. \end{aligned} \tag{6.21}$$

where the slack variable ξ_i measures the margin violation by each data point $\mathbf{x}_i$ and $C \geq 0$ is a penalty parameter for this violation. The original SVM does not have slack variables which were introduced by Cortes and Vapnik (1995). The soft margin version of SVM is very useful when the data is not lineally separable which is the case in many real-world problems. Note that choosing $C = \infty$ retains the hard margin version. The optimisation problem setting is called 1-norm soft margin because of

$$\|\boldsymbol{\xi}\|_1 = \sum_{i=1}^{l}\xi_i. \tag{6.22}$$

The 2-norm soft margin can adopted by minimising

$$\|\boldsymbol{\xi}\|_2 = \sum_{i=1}^{l}\xi_i^2. \tag{6.23}$$

Introducing Lagrange multipliers $\boldsymbol{\alpha}$ to the primal form (6.21), the hyperplane can be found in the dual representation as follows:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximise}} \quad & \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T\mathbf{x}_j, \\ \text{subject to} \quad & \sum_{i=1}^{l} y_i\alpha_i = 0, \\ & C \geq \alpha_i \geq 0\,, \forall i, \end{aligned} \tag{6.24}$$

where the norm of $\mathbf{w}$ is realised by the term:

$$\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j. \tag{6.25}$$

Note that the data appear as an inner product $\mathbf{x}_i^T\mathbf{x}_j$ which means the dual form does not work explicitly on the feature space but on a single value. Hence, the problem complexity is independent from the data dimensionality. The remarkable advantage is to work on high dimensional feature space where the data might be lineally separable without having to compute the mapping explicitly. This is called the kernel trick where a function maps that space to the inner product as follows:

$$K(\boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)) = \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_j). \tag{6.26}$$

There are many nonlinear functions such as Gaussian kernel and polynomial kernel. Although nonlinear SVM could learn complex patterns but it is computationally expensive for large-scale data since the optimisation problem is a quadratic.

One of the early work for fast optimisation is kernel-Adatron algorithm based on a gradient ascent technique (Friess et al., 1998). Decomposition method is also a good technique and the idea is to update a subset of $\boldsymbol{\alpha}$ while keeping the rest constant. Then a new α_i is added to the active set while another old one is removed. Sequential Minimal Optimisation (SMO) is an extreme algorithm works only on two points at each iteration (Platt, 1999).

In the case of linear kernel as in (6.24), the problem is still quadratic but one can take the advantage of accessing the feature space in order to accelerate the optimisation. One of the techniques is a cutting plane solver SVM^{pref} (Joachims, 2006). It is several times faster than decomposition methods such as $\text{SVM}^{\text{light}}$ proposed earlier by the same author. Pegasos is also a fast solver alternates between stochastic gradient steps and projection steps to estimate the primal problem (Shalev-Shwartz et al., 2007). Another technique is a dual coordinate descent method with a shrinking heuristic (Hsieh et al., 2008).

All these techniques can successfully handle large-scale data sets when they are stored in the memory. This means when the resources are limited, the algorithms will take very long time due to severe disk-swapping. Recent work by Yu et al. (2012) addressed this issue and proposed a block minimisation method. The main idea is to divide the data set into m blocks where each block can fit in memory. Then exactly or approximately solve the sub-problem by any of the previous algorithms and update the variables. After going through all the blocks, the optimisation procedure is repeated several times until stopping criterion is satisfied.

We propose a shrinking SVM which is closely related to Hsieh et al. (2008) and Yu et al. (2012). We use dual coordinate ascent method but with a harsh shrinking heuristic to reduce the data points as soon as possible from the first pass. Then the remaining active set is stored into a binary file and a block optimisation is carried out. At each iteration, the active set is reduced until convergence. The key difference between our method and Yu et al. (2012) is that the block optimisation comes after applying harsh shrinking by one pass over the data set. In many cases, the remaining active set is a small fraction of the original set (e.g. 10%) so the block optimisation might not be necessary. Even if this is not the case, dividing the remaining active set is much cheaper than the whole data as we will show in Chapter 7. In addition to that, number of blocks are dynamic and depend on the active set size at each iteration while they are fixed on Yu et al. (2012).

6.3.1 Memory-efficient Stochastic Gradient Method

Gradient method is a simple optimisation algorithm to find a local minimum or maximum of a function using its gradient. Our dual problem (6.24) is convex; hence the solution is unique (Cristianini and Shawe-Taylor, 2000). We start with initial values for $\boldsymbol{\alpha}$ and then move in the direction of the objective function's gradient. This steepest ascent algorithm will reach the maximum point if the learning rate was designed carefully. The partial derivative of (6.24) with respect to one variable is:

$$\frac{\partial W(\boldsymbol{\alpha})}{\partial \alpha_i} = 1 - y_i \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{6.27}$$

and the update rule is:

$$\alpha_i \leftarrow \alpha_i + \eta_i \frac{\partial W(\boldsymbol{\alpha})}{\partial \alpha_i}, \tag{6.28}$$

where η_i is a learning rate. A principled gradient ascent is to update the variables after one complete iteration over the whole data points. However, a fast strategy for convergence is to update them once they have been obtained which is called stochastic gradient ascent. Also a good practice is to present the data points in random order each iteration.

The constraints in the dual problem (6.24) restrict the feasible region and have not been considered by the update rule. For the inequality constraints $C \geq \alpha_i \geq 0$, one can ensure that α_i does not leave the box by the following (Cristianini and Shawe-Taylor, 2000):

$$\alpha_i \leftarrow \min(C, \max(0, \alpha_i + \eta_i \frac{\partial W(\boldsymbol{\alpha})}{\partial \alpha_i})). \tag{6.29}$$

The remaining linear constraints $\sum_{i=1}^{l} y_i \alpha_i = 0$ caused by the bias b can be ignored. The solution might be not optimal but we can represent the bias inside the input by adding one extra dimension to each data point, in which the new vector $\hat{\mathbf{x}} = (\mathbf{x}, \tau)$ and the kernel is computed as:

$$K(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j) = K(\mathbf{x}_i, \mathbf{x}_j) + \tau^2. \tag{6.30}$$

The value of τ can take any arbitrary number and some researchers choose $\tau = 1$ as Hsieh et al. (2008) presented. Cristianini and Shawe-Taylor (2000) show in their book an elegant proof that a safe choice of τ is the radius of a ball contains the whole data points which we can calculate as:

$$R = \max_{1 \leq i \leq l} \|\mathbf{x}_i\|. \tag{6.31}$$

They also show that the maximal gain during iterations is made by choosing:

$$\eta_i = \frac{1}{K(\mathbf{x}_i, \mathbf{x}_i)} \tag{6.32}$$

Algorithm 6 presents the learning procedure known as kernel-Adatron algorithm. The early version was introduced by Friess et al. (1998).

Algorithm 6 Kernel-Adatron algorithm for sovlving 1-norm soft margin SVM

Require: training set S and $C \geq 0$

1: $\boldsymbol{\alpha} = \mathbf{0}$
2: **repeat**
3: Randomly shuffle S
4: **for** $i = 1$ **to** l **do**
5: $G = 1 - y_i \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j)$
6: $\eta = 1/K(\mathbf{x}_i, \mathbf{x}_i)$
7: $\alpha_i \leftarrow \alpha_i + \eta \cdot G$
8: **if** $\alpha_i < 0$ **then**
9: $\alpha_i \leftarrow 0$
10: **else if** $\alpha_i > C$ **then**
11: $\alpha_i \leftarrow C$
12: **end if**
13: **end for**
14: **until** $\boldsymbol{\alpha}$ converge

6.3.2 Linear Kernel

Although kernel-Adatron is simple and fast, computing the gradient (line 5 in Algorithm 6) is expensive for large-scale data. It requires for one update sum over all nonzero

α_i (i.e. active set) multiplied by the corresponding kernel. In the case of linear kernel, we have access to the feature space. Taking this advantage, we can rearrange the gradient as follows:

$$\frac{\partial W(\boldsymbol{\alpha})}{\partial \alpha_i} = 1 - y_i \mathbf{x}_i^T \sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j. \tag{6.33}$$

Now we can notice that the summation part remains the same after each update except one variable. A trick proposed by Hsieh et al. (2008) is to store the summation in a vector $\mathbf{w}$ and when there is an update we replace the old α_i with the new one as shown below:

$$\begin{aligned}
\mathbf{w}_{\text{new}} &= \alpha_1 y_1 \mathbf{x}_1 + \cdots + \alpha_{\text{new}} y_i \mathbf{x}_i + \cdots + \alpha_l y_l \mathbf{x}_l \\
&= \alpha_1 y_1 \mathbf{x}_1 + \cdots + \alpha_{\text{old}} y_i \mathbf{x}_i + \cdots + \alpha_l y_l \mathbf{x}_l \\
&\quad + \alpha_{\text{new}} y_i \mathbf{x}_i - \alpha_{\text{old}} y_i \mathbf{x}_i \\
&= \mathbf{w}_{\text{old}} + (\alpha_{\text{new}} - \alpha_{\text{old}}) y_i \mathbf{x}_i.
\end{aligned} \tag{6.34}$$

This simple trick accelerates kernel-Adatron algorithm for linear SVM and allowed for solving large-scale problem (Hsieh et al., 2008). Algorithm 7 presents the algorithm.

Algorithm 7 Dual coordinate ascent algorithm for 1-norm soft margin linear SVM

Require: training set S and $C \geq 0$

$\boldsymbol{\alpha} = \mathbf{0}$ and $\mathbf{w} = \mathbf{0}$
repeat
    Randomly shuffle S
    for $i = 1$ **to** l **do**
        $\alpha_{\text{old}} \leftarrow \alpha_i$
        $G = 1 - y_i \mathbf{w}^T \mathbf{x}_i$
        $\eta = 1/\mathbf{x}_i^T \mathbf{x}_i$
        $\alpha_i \leftarrow \alpha_i + \eta \cdot G$
        if $\alpha_i < 0$ **then**
            $\alpha_i \leftarrow 0$
        else if $\alpha_i > C$ **then**
            $\alpha_i \leftarrow C$
        end if
        $\mathbf{w} \leftarrow \mathbf{w} + (\alpha_i - \alpha_{\text{old}}) y_i \mathbf{x}_i$
    end for
until $\boldsymbol{\alpha}$ converge

6.3.3 Shrinking Heuristic

The dual coordinate ascent algorithm can converge faster by reducing the size of the problem (i.e. inner loop in Algorithm 7). Such technique is known as decomposition or shrinking. When the data set is large, shrinking becomes important and it is vital when the problem requires more memory than the available.

Hsieh et al. (2008) proposed a shrinking technique which requires to check the old gradient of the objective function before removing any data point. They show that their algorithm terminates after finite iterations. In practice, the removing condition might not be satisfied in the early iterations. When the data cannot fit in memory, there is a need for disk-swapping.

We propose a shrinking heuristic from the first iteration. Less informative data points are removed from memory as soon as possible. Hence, there is only one pass over the whole data. Since the stochastic gradient ascent update $\boldsymbol{\alpha}$ once they have been obtained, any variable $\alpha_i = 0$ has small chance to be positive again. Therefore, the corresponding data point is removed from memory. This technique might be harsh but it is practical for large-scale data. Surprisingly, the achieved accuracy is very close to state-of-the-art solvers as we will show in Chapter 7. The remaining active set (i.e. $\alpha_i > 0$) is stored in a binary file rather than text for efficiency. Then we load part of the file determined by the available memory. Each loaded sub-problem is optimised for N inner iterations to reduce the outer iterations. Hence, the disk access times are limited. Algorithm 8 describes the algorithm.

6.3.4 Kernel Mapping via Linear SVM

We have seen in the previous section that linear SVM can be scalable because the advantage of accessing the feature space. On the other hand, kernel SVM is able to learn more complex patterns by working on high dimensional feature space, where the data might be lineally separable, without explicit mapping using the kernel trick.

An interesting technique to accelerate kernel SVM is to apply linear SVM to the explicit form. However, in many cases, kernel mapping is exponential to the input space or infinite as in Gaussian kernel. Low-degree polynomial mapping is shown to be useful for certain data sets (Chang et al., 2010). A simple example of degree-2 polynomial mapping of $\mathbf{x} \in R^2$ is:

$$\boldsymbol{\phi}(\mathbf{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]$$

All-subsets kernel is similar to polynomial but has more flexibility in terms of the monomials' weightings (Shawe-Taylor and Cristianini, 2004). It is defined by the embedding

Algorithm 8 Memory-efficient dual coordinate ascent algorithm for linear SVM

Require: training set S , $C \geq 0$ and $N \geq 1$

1: $\boldsymbol{\alpha} = \mathbf{0}$ and $\mathbf{w} = \mathbf{0}$
2: Open a binary file BF
3: **for** $i = 1$ **to** l **do**
4: Do from line 5 to 14 in Algorithm 7
5: **if** $\alpha_i > 0$ **then**
6: Store $\mathbf{x}_i, y_i$ in BF
7: **end if**
8: Remove $\mathbf{x}_i, y_i$ from memory
9: **end for**
10: **repeat**
11: $i \leftarrow 1$
12: **repeat**
13: **if** $\alpha_i > 0$ **then**
14: Read $\mathbf{x}_i, y_i$ from BF
15: $i \leftarrow i + 1$
16: **end if**
17: **if** memory is full **then**
18: Do lines (3-15) in Algorithm 7 for N times
19: Remove all data points from memory
20: **end if**
21: **until** end of BF
22: **until** $\boldsymbol{\alpha}$ converge

$$\boldsymbol{\phi} : \mathbf{x} \mapsto (\boldsymbol{\phi}_A(\mathbf{x}))_{A \subseteq \{1,\ldots,n\}}, \tag{6.35}$$

where the feature $\boldsymbol{\phi}_A$ for each subset A is given by

$$\boldsymbol{\phi}_A(\mathbf{x}) = \prod_{i \in A} x_i. \tag{6.36}$$

The mapping generates all combinations of input features and each monomial's coefficient equals one unlike polynomial mapping. A mapping example of $\mathbf{x} \in R^3$ is:

$$\phi(\mathbf{x}) = [x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1x_2x_3]$$

Working with all monomials might be computationally expensive. ANOVA kernel K_d restricts the mapping to subsets of cardinality d with $\binom{n}{d}$ dimensions. The embedding is given by

$$\boldsymbol{\phi} : \mathbf{x} \mapsto (\boldsymbol{\phi}_A(\mathbf{x}))_{|A|=d}. \tag{6.37}$$

6.4 Multiclass Support Vector Machine

Phrase reordering problem is usually formulated as multiclass problem which can be solved as several binary problems in SVM. One-versus-rest or one-versus-one are well known strategies. The former requires K classifiers while the latter requires $K(K-1)/2$ classifiers. We choose the one-versus-rest strategy (Algorithm 9) because it is simpler to implement and the number of the weight vectors $\mathbf{w}_k$ is smaller. Algorithm 9 has two advantages. First, there is only one pass over the whole data to filter out easily classified points in each binary problem. This step saves both memory and time. Second, each binary problem is independent from the other. Hence, we might have K threads and run the algorithm in parallel. This step will speed the process a lot but our experiments are based on a single thread.

Algorithm 9 Memory-efficient one-versus-rest strategy for multiclass SVM

Require: training set S , $C \geq 0$, $N \geq 1$ and $K \geq 3$

1: $\boldsymbol{\alpha}_k = \mathbf{0}$ and $\mathbf{w}_k = \mathbf{0}$
2: Open K binary files BF_k
3: **for** $i = 1$ **to** l **do**
4: **for** $k = 1$ **to** K **do**
5: **if** $y_i = k$ **then**
6: $y \leftarrow +1$
7: **else**
8: $y \leftarrow -1$
9: **end if**
10: Do from line 5 to 14 in Figure 7
11: **if** $\alpha_i > 0$ **then**
12: Store $\mathbf{x}_i, y$ in BF_k
13: **end if**
14: Remove $\mathbf{x}_i, y_i$ from memory
15: **end for**
16: **end for**
17: For each BF_k do from line 10 to 22 in Algorithm 8

A more elegant way for multi-class problem is to construct a single classifier where the margin is maximised between all classes simultaneously as shown in case (c) Figure 6.1. Crammer and Singer (2001) introduce a multiclass SVM formulation by solving the following optimisation problem:

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimise}} \quad & \frac{1}{2}\sum_{k=1}^{K} \mathbf{w}_k^T \mathbf{w}_k + C\sum_{i=1}^{N} \xi_i, \\ \text{subject to} \quad & \mathbf{w}_{y_i}^T \mathbf{x}_i + \delta_{ik} - \mathbf{w}_k^T \mathbf{x}_i \geq 1 - \xi_i \,, \forall i, \\ & \delta_{ik} = \begin{cases} 0 & \text{if } y_i \neq y_k; \\ 1 & \text{if } y_i = y_k. \end{cases} \end{aligned} \tag{6.38}$$

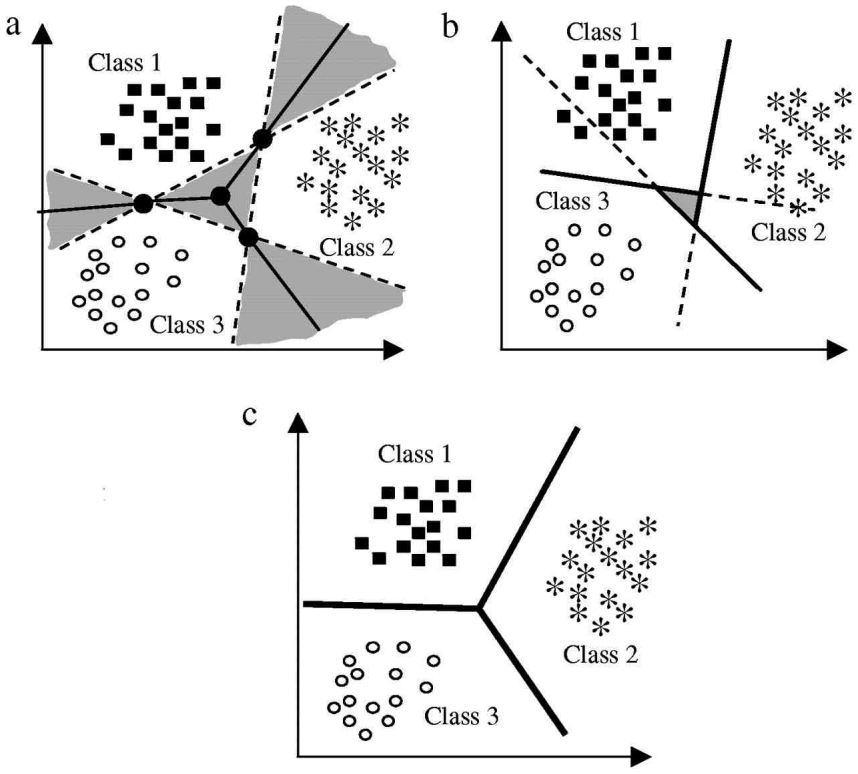

FIGURE 6.1: Three SVM approaches for multi-class problem: (a) one-versus-rest (b) one-versus-one (c) multi-class SVM [taken from Statnikov et al. (2005)]

The dual problem is:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimise}} \quad & \mathcal{D}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j=1}^{N}\alpha_{ik}\alpha_{jk}\mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{N}\sum_{k=1}^{K}(1-\delta_{ik})\alpha_i, \\ \text{subject to} \quad & \sum_{k=1}^{K}\alpha_{ik} = 0 \quad \text{and} \quad \alpha_{ik} \leq C\delta_{ik} \quad \forall i,k. \end{aligned} \tag{6.39}$$

where the corresponding $\mathbf{w}_k$ defined as:

$$\mathbf{w}_k = \sum_{i=1}^{N}\alpha_{ik}\mathbf{x}_i \tag{6.40}$$

Given a new sample $\mathbf{x}$, the decision function is:

$$\underset{k}{\text{argmax}}\sum_{i=1}^{N}\alpha_{ik}{\mathbf{x}_i}^T\mathbf{x} \quad \equiv \quad \underset{k}{\text{argmax}}\,\mathbf{w}_k^T\mathbf{x} \tag{6.41}$$

In our experiments we used softmax function to yield a probabilistic decision. Note that a bias parameter b_k is augmented in $\mathbf{w}_k$.

Keerthi et al. (2008) propose a sequential dual method to solve the problem (6.39). The method sequentially picks $\mathbf{x}_i$ at a time and optimises its dual variable (i.e. $\alpha_{ik}\ \forall k$) while fixing all other variables. The sub-problem is given by:

$$\begin{aligned} \underset{\boldsymbol{\alpha}_i}{\text{minimise}} \quad & \frac{1}{2}\sum_{k=1}^{K}\frac{1}{2}A\alpha_{ik}^2 + B_k\alpha_{ik}, \\ \text{subject to} \quad & \alpha_{ik} \leq C\delta_{ik} \quad \forall k, \end{aligned} \tag{6.42}$$

where

$$\begin{aligned} A = \mathbf{x}_i^T\mathbf{x}_i \quad \text{and} \quad B_k = G_{ik} - A\alpha_{ik}, \\ G_{ik} = \frac{\partial\mathcal{D}(\boldsymbol{\alpha})}{\partial\alpha_{ik}} = \mathbf{w}_k^T\mathbf{x}_i + 1 - \delta_{ik}. \end{aligned} \tag{6.43}$$

Crammer and Singer (2001) provide $O(k\log k)$ algorithm to solve the sub-problem (6.42). Fan et al. (2008) present a simpler version given in Algorithm 10.

Algorithm 10 Solving the sub-problem of multiclass SVM

Require: A, $\mathbf{B}$ and a penalty parameter $C \geq 0$

1: $D_k \leftarrow B_k + AC\delta_{ik}$, $\forall k$
2: Sort $\mathbf{D}$ in decreasing order
3: $\beta \leftarrow D_1 - AC$
4: $r \leftarrow 2$
5: **while** $r \leq K$ and $\beta/(r-1) < D_r$ **do**
6: $\quad\beta \leftarrow \beta + D_r$
7: $\quad r \leftarrow r + 1$
8: **end while**
9: $\beta \leftarrow \beta/(r-1)$
10: $\alpha_{ik}^{'} \leftarrow min(C\delta_{ik}, (\beta - B_k)/A)$, $\forall k$

After each update, the corresponding weight vector for each class $\mathbf{w}_k$ is changed as follows:

$$\mathbf{w}_k = \mathbf{w}_k + (\alpha_{ik}^{'} - \alpha_{ik})\mathbf{x}_i \tag{6.44}$$

The optimal dual variables are achieved when the following condition is satisfied for all samples (Keerthi et al., 2008):

$$v_i = 0\,, \forall i, \tag{6.45}$$

where

$$v_i = \max_k G_{ik} - \min_{k:\alpha_{ik}<C\delta_{ik}} G_{ik}. \tag{6.46}$$

This condition is a key to accelerate the algorithm. Unlike the primal problem (6.38), the dual variables α_{ik} are associated with each sample (i.e. phrase pair) therefore a training sample can be disregarded once its optimal dual variables are obtained. More data shrinking can be achieved by tolerating a small difference between the two values in (6.46). Algorithm 11 presents the overall procedure.

Algorithm 11 Shrinking dual method for training large-scale multiclass SVM

Require: training set $S = \{\mathbf{x}_i, y_i\}_{i=1}^N$

$\boldsymbol{\alpha} = \mathbf{0}$ and $\mathbf{w} = \mathbf{0}$
repeat
    Randomly pick i from S
    Claculate A_{ik} , B_{ik} , G_{ik} $\forall k$ by (6.43)
    Claculate v_i by (6.46)
    if $v_i \leq \epsilon$ **then**
        Remove i from S
    else
        Calculate α'_{ik} $\forall k$ by Algorithm 10
        Update $\boldsymbol{\alpha}$ and $\mathbf{w}$ by (6.44)
    end if
until $v_i \leq \epsilon$ $\forall i$

6.5 Chapter Summary

In this chapter, we proposed a variety of large-scale classifiers from generative models to discriminative ones. In the next chapter, we evaluate these classification models along with widely used models disscused in Chapter 5. This evaluation will be in two phases. First, the classifiers are examined on benchmark datasets in the field of pattern recognition in general. Second, reordering models are built based on these classifiers from a parallel corpus.

Chapter 7

Classification Experiments and Results

Posing phrase movements as a classification problem, we exploit various classifiers throughout Chapter 5 and 6. This chapter evaluates them in two stages. First, the classifiers are examined on benchmark datasets in the field of pattern recognition in general. Second, we examine the ability of these classifiers in predicting phrase reordering using two parallel corpora.

7.1 Pattern Recognition in a Variety of Datasets

Seven benchmark datasets have been used in our experiments, six of them are available at LIBSVM webpage[1]. One of the datasets namely URL was introduced by Ma et al. (2009) for detecting malicious web sites. Four datasets are from text classification (RCV1 Lewis et al. (2004), WEBSPAM Webb et al. (2006), NEWS20 Lang. (1995) and SECTOR McCallum and Nigam (1998)). We also used an artificial dataset from PASCAL Challenge 2008[2] (EPSILON). A small and large handwritten digits sets were also tested (MNIST Lecun et al. (1998) and MNIST8M Loosli et al. (2007)). The last dataset is from KDD archive[3] for detecting legitimate connection in a computer network. Table 7.1 shows the training data statistics for each set. The test size for each one is: 239,613 (URL), 20,242 (RCV1), 50,000 (WEBSPAM), 62,060 (NEWS20), 3,207 (SECTOR), 100,000 (EPSILON), 10,000 (MNIST and MNIST8M) and 311,029 (KDD 1999).

[1] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets
[2] http://largescale.ml.tu-berlin.de
[3] http://kdd.ics.uci.edu

Dataset	# class	l	n	# nonzeros
URL	2	2,156,517	3,231,961	249,347,283
RCV1	2	677,399	47,236	49,556,258
RCV1.multi	53	518,571	47,236	33,486,015
WEBSPAM	2	300,000	16,609,143	1,117,924,883
NEWS20	20	15,935	62,061	1,272,569
SECTOR	105	6,412	55,197	1,045,412
EPSILON	2	400,000	2,000	800,000,000
MNIST	10	60,000	784	8,994,156
MNIST8M	10	8,100,000	784	1,612,242,143
KDD 1999	2	4,898,431	41	61,138,555

TABLE 7.1: Statistics of seven benchmark datasets used in our experiments (l is the number of instances and n is the number of features).

7.1.1 Results of Support Vector Machine

We compare our shrinking SVM with three state-of-the-art linear SVM solvers. Each one maximises 1-norm soft margin SVM but approaches the problem with different method. The first solver is LIBLINEAR[4] version 1.94, which is a dual coordinate descent method with a shrinking heuristic (Hsieh et al., 2008). We used it with options '`-s 3 -B 1`'. The second solver is Pegasos[5] which estimates the primal problem with stochastic sub-gradient descent algorithm (Shalev-Shwartz et al., 2007). The last one is SVM$^{\text{pref}}$ version 3.00[6], which is a cutting plane method solver (Joachims, 2006).

	LLSVM (our method)		LIBLINEAR		Pegasos		SVM$^{\text{pref}}$	
Dataset	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy
URL	2m19s	99.54%	6m21s	99.59%	1m56s	97.51%	15m4s	99.18%
RCV1	0m22s	97.67%	0m25s	97.78%	0m46s	96.03%	1m9s	97.72%
WEBSPAM	10m44s	99.00%	(2m43s)	(98.30%)	N/A	N/A	N/A	N/A
EPSILON	5m29s	89.75%	(1m48s)	(89.30%)	(4m10s)	(82.91%)	(3m5s)	(89.29%)
MNIST	0m33s	92.07%	0m41s	92.18%	N/A	N/A	N/A	N/A
MNIST8M	174m25s	90.86%	(50m28s)	(88.95%)	N/A	N/A	N/A	N/A
KDD 1999	0m30s	92.29%	0m37s	92.05%	0m41s	86.80%	1m31s	92.07%

TABLE 7.2: Comparison between our method and three popular SVM solvers (LIBLINEAR, Pegasos, SVM$^{\text{pref}}$) on several datasets. The reported elapsed real time is in minutes (m) and seconds (s). Fields with brackets ( ) means the result is based on 25% of the data due to memory restriction in our machine (12 GB RAM). Some fields are not available (i.e. N/A) because the software does not solve multiclass problem internally.

All solvers are implemented in C/C++ including ours. Our code will be available on GitHub website [7]. The penalty parameter C is set to 1 following Hsieh et al. (2008) (other values have no significant gain in our experiments). The equivalent setting for Pegasos parameter λ is $1/(Cl)$ and for SVM$^{\text{pref}}$ parameter C_{pref} is $0.01Cl$. The following relative

[4] http://www.csie.ntu.edu.tw/∼cjlin/liblinear
[5] http://ttic.uchicago.edu/∼shai/code
[6] http://svmlight.joachims.org/svm_perf.html
[7] https://github.com/asrajeh

Dataset	LLSVM (our method)	LIBLINEAR	Mismatch
URL	2,048,691	2,018,715	0.0441
RCV1	665,002	662,902	0.0183
MNIST d8	57,955	55,910	0.0531
KDD 1999	4,886,751	4,893,975	0.0023

TABLE 7.3: Non-active dual variables in our method compared to LIBLINEAR and the fraction of variables we mismatched.

Data Set	Split	Train	Total Time	Accuracy
WEBSPAM	19m12s	4m48	24m0s	99.04 %
EPSILON	14m10s	3m26	17m36s	89.75 %

TABLE 7.4: Block minimisation for large-scale problems (Yu et al., 2012).

difference function between $W(\boldsymbol{\alpha}*)$ (current iteration) and $W(\boldsymbol{\alpha})$ (previous iteration) is the stopping criterion defined as:

$$\frac{W(\boldsymbol{\alpha}*) - W(\boldsymbol{\alpha})}{W(\boldsymbol{\alpha}*)} \leq 0.01 \tag{7.1}$$

There are three observations we can state about the results in Table 7.2. First, our solver (LLSVM) is always faster except in the first data set (URL) where Pegasos is faster. However, it has relatively low test accuracy 97.51% compared to ours 99.53%. In fact, Pegasos always has the lowest accuracy and consumes more memory in our experiments. Second, we achieve comparable accuracy with state-of-the-art solvers and sometimes higher. Third, the proposed method could solve very large problem such as WEBSPAM, EPSILON and MNIST8M with reasonable time.

Shrinking the training data early is the main reason for the proposed method's achievement. Figure 7.1 and Figure 7.2 show the percentage of active set for the first 10 iterations in our solver compared with LIBLINEAR. The amount of reduction for some data set is very huge due to the problem nature. For example, almost 99% of KDD is removed from the first iteration.

Although the adopted harsh shrinking method reduced our objective function value (6.24) as illustrated in Figure 7.3, it seems to have little impact on the test accuracy while it largely saves memory and time. Non-active dual variables we found from early stage are surprisingly very similar to LIBLINEAR's solution as Table 7.3 shows. In Figure 7.2, LIBLINEAR takes several iterations to discover the active set.

Block minimisation proposed by Yu et al. (2012) is an alternative method to ours. Table 7.4 reports training time and accuracy for two large-scale problems. It took 19 minutes to just split WEBSPAM and 14 minutes to split EPSILON while in Table 7.2 the whole training time is about 10 minutes (nearly half) and 5 minutes (nearly third), respectively. Although we do block minimisation when a data set after shrinking cannot

fit in memory however dividing the remaining active set is much cheaper than the whole data.

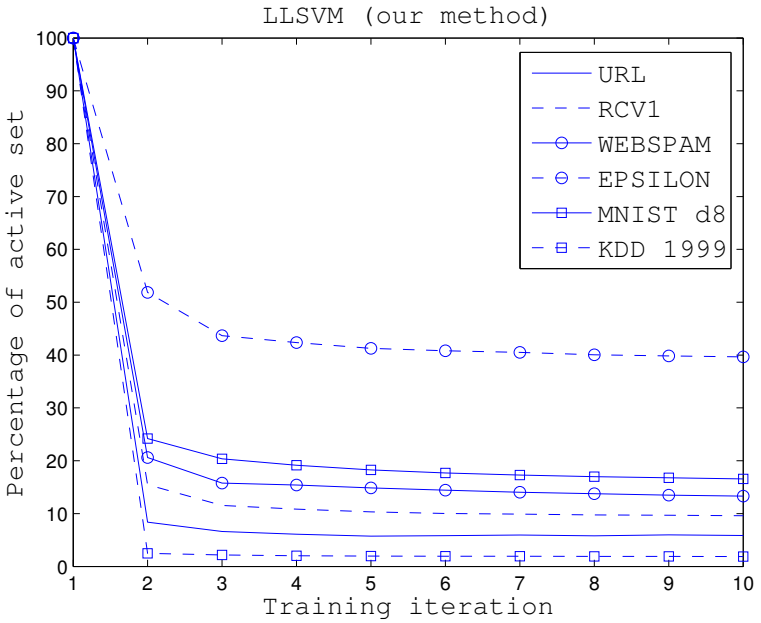

FIGURE 7.1: Percentage of active set for 10 iterations in our solver (d8 is digit 8 versus all).

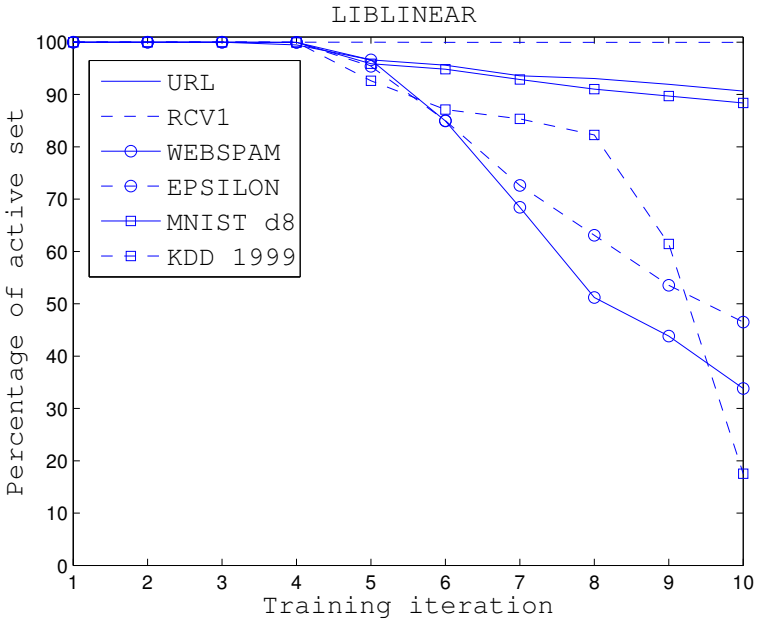

FIGURE 7.2: Percentage of active set for 10 iterations in LIBLINEAR (d8 is digit 8 versus all).

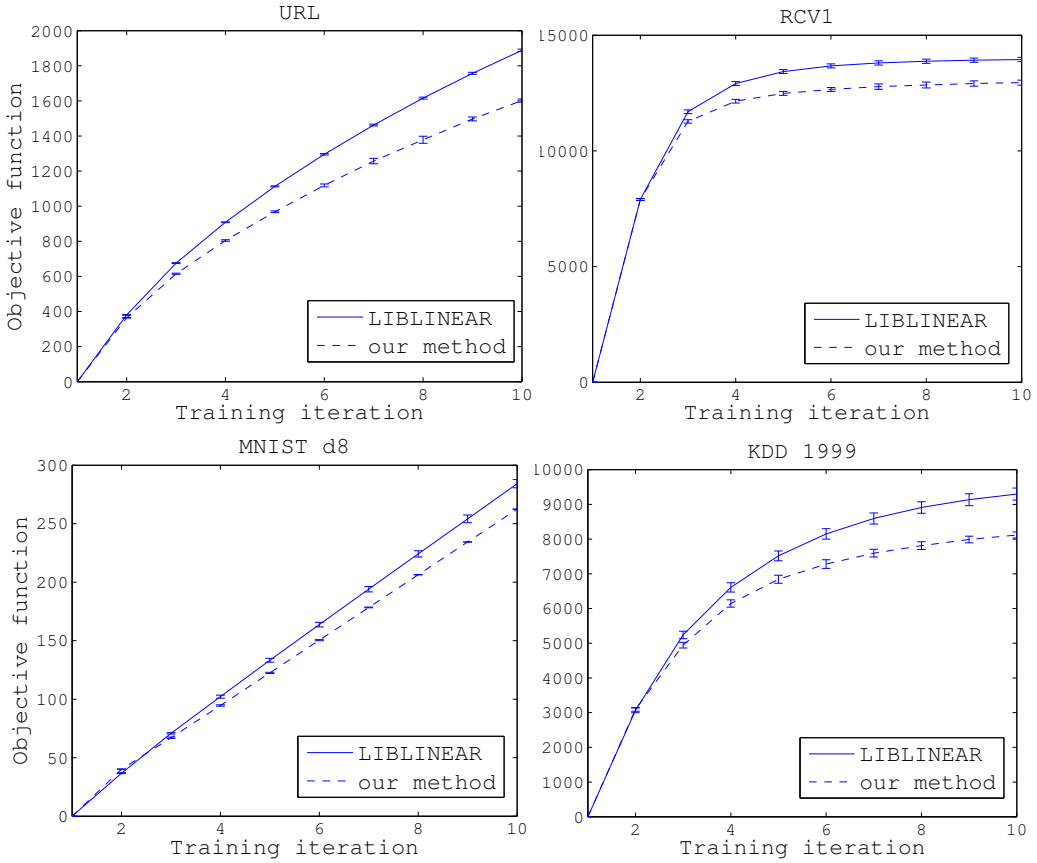

FIGURE 7.3: Function value (6.24) of each data set for 10 iterations.

7.1.2 Results of Multiclass SVM and Logistic Regression

In the previous section, we presented the results of a binary classifier (linear SVM). Here, we show the results of multiclass SVM and also multinomial logistic regression in the primal and dual problems.

Muliclass SVM usually has higher accuracy than the competing classifiers as shown in Table 7.5. It is also faster than dual multinomial logistic regression (MLR) although the latter has a shrinking mechanism. Figure 7.4 explains the reason and shows that muliclass SVM ables to shrink faster.

	LLSVM		Multiclass SVM		Primal MLR		Dual MLR	
Dataset	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy
URL	2m19s	99.5%	1m19s	99.2%	6m52s	97.5%	4m32s	99.3%
RCV1.multi	2m15s	92.0%	1m25s	92.1%	31m06s	90.2%	12m46s	92.1%
NEWS20	0m3s	85.3%	0m3s	85.4%	0m29s	85.7%	0m12s	84.2%
SECTOR	0m9s	94.1%	0m9s	94.5%	2m3s	92.8%	0m28s	92.7%
MNIST	0m33s	92.1%	0m42s	92.8%	1m14s	92.4%	0m36s	92.6%

TABLE 7.5: Comparison between different classifiers on several muliclass datasets. The reported time is in minutes (m) and seconds (s).

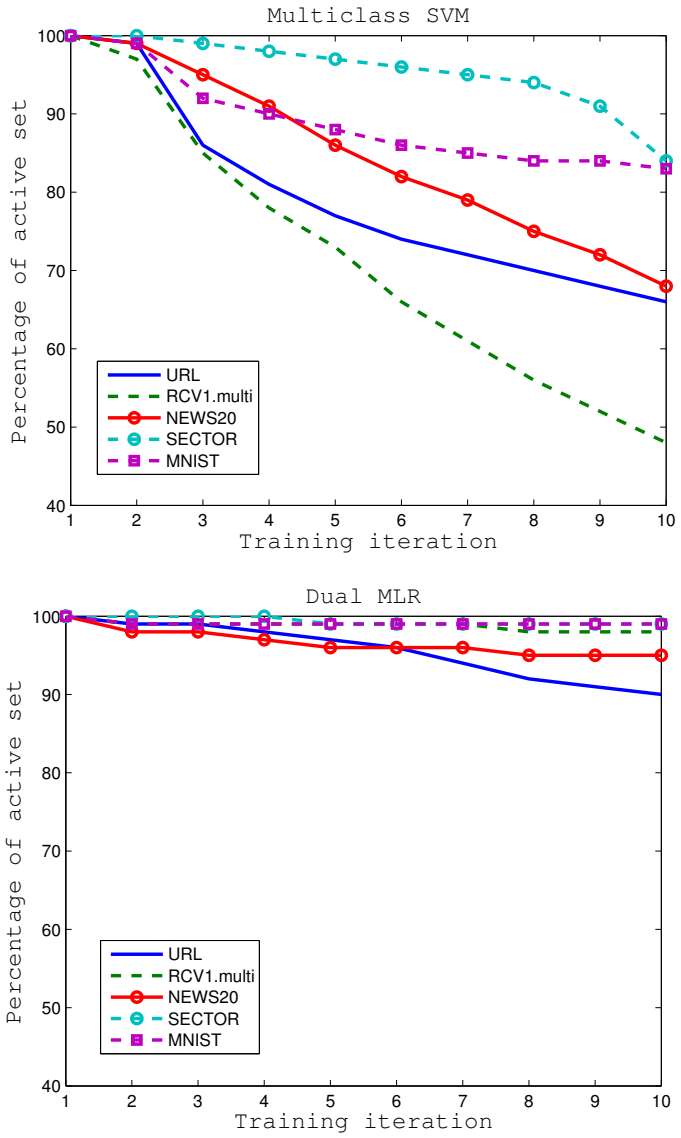

FIGURE 7.4: Percentage of active set for 10 iterations in Multiclass SVM (above) and in Dual Multinomial Logistic Regression (below).

7.1.3 Results of Multinomial Naive Bayes

We discussed in Section 6.1, naive Bayes can be estimated by MAP or Bayesian inference. The results shows that there is no advantage to using Bayesian inference instead of MAP estimate as shown in Table 7.6.

	LLSVM		MAP Naive Bayes		Bayesian Naive Bayes	
Dataset	Time	Accuracy	Time	Accuracy	Time	Accuracy
NEWS20	0m3s	85.3%	0m1s	85.1%	0m1s	85.0%
MNIST	0m33s	92.1%	0m3s	83.8%	0m3s	83.7%
MNIST8M	174m25s	90.9%	7m18s	82.3%	7m18s	82.2%
KDD 1999	0m30s	92.3%	0m19s	90.9%	0m19s	90.7%

TABLE 7.6: Results of MAP and Bayesian Naive Bayes compared to SVM.

One of the reason might be that the posterior distribution of Dirichlet's parameters is very sharp. This makes MAP method a good estimation of the posterior distribution. Figure 7.5 illustrates the probability density function (PDF) of one parameter when we have higher feature space. Having large training data will also results in a sharply peaked posterior (Ghahramani, 2001).

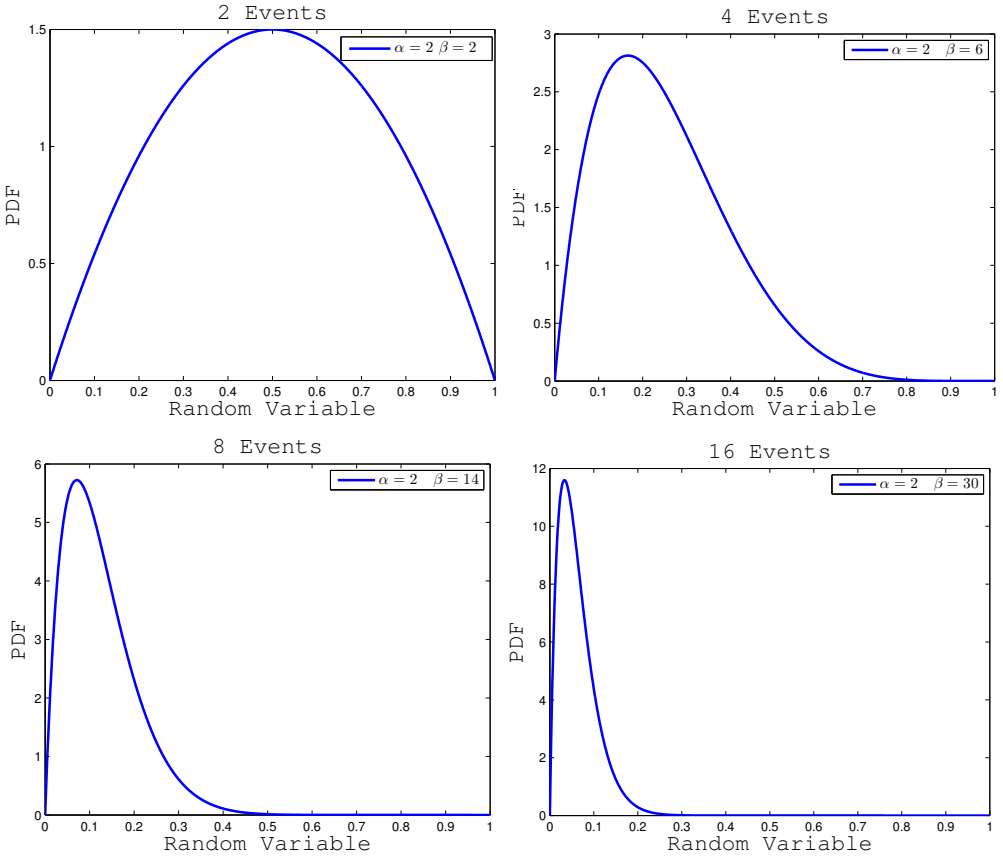

FIGURE 7.5: The probability density function (PDF) of one Dirichlet's parameter when the number of parameters increases.

7.1.4 Results of Voted Spheres

Voted Spheres is a non-linear scalable classifier (Farran and Saunders, 2009). Table 7.7 shows the classifier has higher accuracy than linear SVM however it requires a lot of time to train and more importantly testing time is very slow. It is 100 times or more slower than LIBLINEAR. This makes it not practical for machine translation systems.

	LIBLINEAR			Voted Spheres		
Dataset	Train	Test	Accuracy	Train	Test	Accuracy
MNIST	0m41s	0m1s	92.2%	106m51s	74m26s	93.3%
KDD 1999	0m40s	0m2s	92.1%	31m30s	11m49s	92.5%
KDD 1999 (10%)	0m5s	0m2s	92.2%	1m25s	4m50s	94.3%

TABLE 7.7: Results of Voted Spheres against linear SVM.

7.1.5 Results of Ordinal Regression

Ordinal regression differs from classification in that the output is an ordinal variable. Therefore, we considered different datasets in order to have a fair comparison. We have chosen five benchmark datasets (Chu and Ghahramani, 2005). Table 7.8 presents statistics of these benchmark datasets.

Dataset	n	l (Training)	l (Testing)
Boston Housing	13(12,1)	300	206
Stocks Domain	9(9,0)	600	350
Abalone	8(7,1)	1,000	3,177
Bank Domains	32(32,0)	4,000	4,192
Census Domains	16(16,0)	12,000	10,784

TABLE 7.8: Statistics of five benchmark datasets used in our experiments (l is the number of instances and n is the number of features). (Numeric,Nominal)

For this task, we have used two evaluation metrics:

- Mean absolute accuracy is the average deviation of the output from the true class;

$$\text{Absolute Accuracy} = \frac{1}{N}\sum_{i=1}^{N} 1 - \frac{|y_i - y_i^*|}{k-1}. \tag{7.2}$$

- Mean zero-one accuracy gives 1 to every correct output.

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \delta_{y_i y_i^*} \quad , \quad \delta_{y_i y_i^*} = \begin{cases} 0 & \text{if } y_i \neq y_i^*; \\ 1 & \text{if } y_i = y_i^*. \end{cases} \tag{7.3}$$

We compared ordinal regression with maximum entropy classifier and variable margin perceptron (discussed in Section 5.3). Ni et al. (2011) proposed a multiclass perceptron with a distance function $\Delta(y_i, y_k)$ between classes. This classifier is presented as a better approach to model ordinal data than a standard multiclass perceptron. The three classifiers' results are tested on benchmark datasets discretised by 5 and 10 equal-length bins as shown in Table 7.9 and Table 7.10, respectively.

	Zero-One Accuracy			Absolute Accuracy		
Dataset	Perceptron	MaxEnt	OrdReg	Perceptron	MaxEnt	OrdReg
Boston Housing	50.5%	66.5%	**74.7%**	83.4%	90.8%	**92.4%**
Stocks Domain	74.3%	**80.3%**	55.7%	93.1%	**95.0%**	88.5%
Abalone	**78.4%**	78.1%	77.1%	**94.3%**	94.1%	93.8%
Bank Domains	41.1%	**47.4%**	46.8%	77.6%	81.8%	**82.9%**
Census Domains	40.0%	**48.2%**	44.9%	75.5%	81.1%	**81.2%**

TABLE 7.9: Results of ordinal regression, maximum entropy and variable margin perceptron on benchmark datasets discretised by 5 equal-length bins.

Dataset	Zero-One Accuracy			Absolute Accuracy		
	Perceptron	MaxEnt	OrdReg	Perceptron	MaxEnt	OrdReg
Boston Housing	35.4%	**46.6**%	41.2%	88.0%	**91.4**%	89.0%
Stocks Domain	54.8%	**60.0**%	35.1%	94.3%	**94.9**%	90.3%
Abalone	57.2%	**57.3**%	55.3%	**94.1**%	94.0%	93.8%
Bank Domains	21.7%	25.4%	**25.6**%	79.8%	82.2%	**83.8**%
Census Domains	20.8%	**26.5**%	24.2%	78.8%	81.5%	**82.0**%

TABLE 7.10: Results of ordinal regression, maximum entropy and variable margin perceptron on benchmark datasets discretised by 10 equal-length bins.

7.2 Phrase Reordering Modelling

We simplify the problem by classifying phrase movements into three categories (monotone, swap, discontinuous). To train the reordering models, we used GIZA++ to produce word alignments (Och and Ney, 2000). Then, we used the `extract` tool that comes with the Moses [8] toolkit (Koehn et al., 2007) in order to extract phrase pairs along with their orientation classes.

Feature Set	Phrase Pair			Context		ANOVA Mapping		
	all words	boundaries	alignments	size=1	size=3	d=1	d=2	d≤2
S1	•					✓		
S2		•				✓		
S3			•			✓		
S4		•		•		✓		
S5		•		•			✓	
S6		•		•				✓
S7			•	•		✓		
S8			•	•			✓	
S9			•	•				✓
S10		•			•	✓		
S11		•			•		✓	
S12		•			•			✓
S13			•		•	✓		
S14			•		•		✓	
S15			•		•			✓

TABLE 7.11: A variety of feature sets to represent a phrase pair.

During the extraction process, each extracted phrase pair is represented by linguistic features. There are different feature representations in the literature as we have seen in Section 4.1. We explore a variety of feature sets as shown in Table 7.11. Each phrase pair in represented by all its words, its boundaries or its alignments. We have considered one or three words of context (i.e. occur before or after each phrase pair). Finally, one of ANOVA mappings were selected. Table 7.12 gives a generic example.

[8] Moses is an open source toolkit for statistical machine translation (www.statmt.org/moses/).

Sentence pair:

Foreign sentence $\mathbf{f}$: f_1 f_2 $_1$ f_3 f_4 f_5 $_2$ f_6 $_3$.

English sentence $\mathbf{e}$: e_1 $_1$ e_2 e_3 $_3$ e_4 e_5 $_2$.

Extracted phrase pairs $(\bar{\mathbf{f}}, \bar{\mathbf{e}})$:

$\bar{f}_i$	$\bar{e}_i$	o_i	alignments
f_1 f_2	e_1	mono	0-0 1-0
f_3 f_4 f_5	e_4 e_5	swap	0-1 2-0
f_6	e_2 e_3	disc.	0-0 0-1

Feature Representation:
a phrase pair is represented as a vector $\boldsymbol{\phi}$ where each feature is a discrete number (0=not exist). Below is a representation of $\boldsymbol{\phi}(\bar{f}_2, \bar{e}_2)$ in different feature sets:

S1 : f_3, f_4, f_5, e_4, e_5

S2 : f_3, f_5, e_4, e_5

S3 : f_3&e_5, f_5&e_4

S4 : f_3, f_5, e_4, e_5, f_2^-, f_6^+

S5 : f_3-f_5, f_3-e_4, f_3-e_5, f_3-f_2^-, f_3-f_6^+, f_5-e_4, f_5-e_5, f_5-f_2^-, f_5-f_6^+, e_4-e_5, e_4-f_2^-, e_4-f_6^+, f_2^--f_6^+

S6 : f_3, f_5, e_4, e_5, f_2^-, f_6^+, f_3-f_5, f_3-e_4, f_3-e_5, f_3-f_2^-, f_3-f_6^+, f_5-e_4, f_5-e_5, f_5-f_2^-, f_5-f_6^+, e_4-e_5, e_4-f_2^-, e_4-f_6^+, f_2^--f_6^+

TABLE 7.12: A generic example of the process of phrase pair extraction and representation in different feature sets

7.2.1 Results of Arabic to English

Phrase movements is formalised as classification problem where each phrase position considered as a class. As in Moses, we classify phrase movements into three categories (monotone, swap, discontinuous). There are three ways to measure the phrase movements (as discussed in Section 3.1) which are word-based, phrase-based or hierarchical (Galley and Manning, 2008). Figure 7.6 shows phrase pairs' distribution in our Arabic-English corpus (see Section 8.1).

We evaluated reordering models in terms of accuracy and F_1 score. All experiments reported here were repeated three times to evaluate the uncertainties in our results. They all have standard deviation less than 1.

Firstly, we present the performance of lexicalised reordering models in Table 7.13, 7.14 and 7.15. Then, we compare naive Bayes, MaxEnt and multiclass SVM under all feature

sets in Table 7.11. The hierarchical method considered more phrase pairs as monotone which will lead to better modelling as in Table 7.15. Hence, our experiments are based on a hierarchical method.

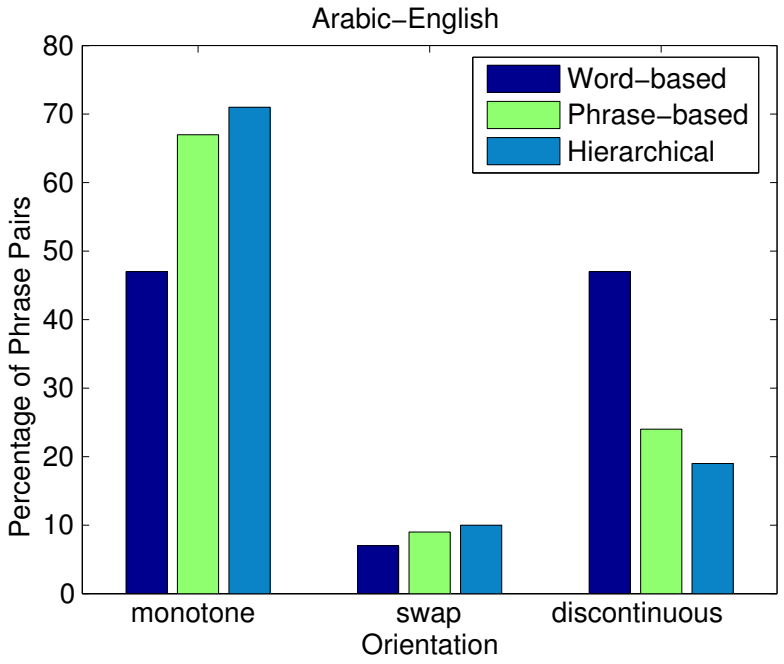

FIGURE 7.6: Distribution of Arabic-English phrase pairs over three orientations estimated by word-based, phrase-based or hierarchical models.

Orientation	Confusion Matrix			Accuracy	F_1 score
	Monotone	Swap	Discontinuous	all classes	
Monotone	41.3	0.5	5.1		66.6
Swap	3.6	1.9	1.6	56.2	37.2
Discontinuous	32.4	0.8	12.9		39.4

TABLE 7.13: The performance of lexicalised reordering model (word-based).

Orientation	Confusion Matrix			Accuracy	F_1 score
	Monotone	Swap	Discontinuous	all classes	
Monotone	64.2	0.8	1.5		82.7
Swap	5.8	2.5	0.9	71.6	38.0
Discontinuous	18.8	0.7	5.0		31.3

TABLE 7.14: The performance of lexicalised reordering model (phrase-based).

Orientation	Confusion Matrix			Accuracy	F_1 score
	Monotone	Swap	Discontinuous	all classes	
Monotone	68.9	0.9	1.3		85.9
Swap	6.4	2.6	0.8	75.9	37.7
Discontinuous	14.2	0.6	4.4		34.5

TABLE 7.15: The performance of lexicalised reordering model (hierarchical).

There are five observations can be drawn from the results in Table 7.16, 7.17 and 7.18. First, naive Bayes achieves good accuracy when we have alignment features (S8). Second, the performance of multiclass SVM is similar to MaxEnt in most feature sets. Third, our classifier is a couple of times faster than MaxEnt (around 4-fold) due to the shrinking method. Forth, context around phrase pairs is important to achieve high accuracy and only one word before and after is enough. Finally, alignment features usually have higher F_1 score than boundary features in the three classifiers.

Feature Set	Train Time	Accuracy	F_1 score		
			Monotone	Swap	Discontinuous
S1	28m	66.9	79.5	28.0	32.1
S2	23m	69.3	81.2	40.2	40.5
S3	28m	73.6	84.5	46.8	46.3
S4	33m	71.2	82.1	48.0	45.2
S5	1h41m	75.2	84.5	56.6	55.0
S6	1h57m	72.9	82.7	54.1	53.5
S7	38m	75.1	85.1	54.2	50.6
S8	1h12m	**77.4**	**86.3**	55.8	54.2
S9	1h29m	75.5	84.9	**57.0**	**56.0**
S10	42m	68.3	79.9	43.0	44.9
S11	4h12m	71.7	81.7	51.8	53.7
S12	4h48m	68.7	78.9	45.1	50.4
S13	48m	72.4	83.3	49.1	49.5
S14	3h18m	74.0	83.8	53.4	55.7
S15	3h41m	72.8	82.8	52.9	55.3

TABLE 7.16: Naive Bayes reordering model's performance. The reported time is in hours (h) and minutes (m).

Feature Set	Train Time	Accuracy	F_1 score		
			Monotone	Swap	Discontinuous
S1	1h26m	74.1	84.8	26.6	29.8
S2	1h10m	74.0	84.7	22.7	29.2
S3	1h40m	76.1	86.1	40.7	43.0
S4	1h50m	77.0	86.4	48.5	39.9
S5	5h59m	80.7	88.7	58.4	53.7
S6	6h21m	81.3	89.1	61.4	55.4
S7	3h10m	78.7	87.6	54.0	48.3
S8	4h32m	81.4	89.3	60.4	57.6
S9	4h43m	82.5	89.9	65.2	59.9
S10	2h45m	76.2	86.0	43.9	40.9
S11	15h11m	82.4	89.7	64.4	57.7
S12	16h04m	82.6	89.5	64.8	58.0
S13	3h24m	78.8	87.9	53.1	51.4
S14	13h03m	82.2	88.4	61.1	54.7
S15	15h12m	**82.9**	**90.8**	**65.7**	**61.4**

TABLE 7.17: Maximum entropy-based reordering model's performance.

Feature Set	Train Time	Accuracy	F_1 score		
			Monotone	Swap	Discontinuous
S1	30m	70.8	83.8	12.0	27.8
S2	28m	71.7	84.2	18.6	17.3
S3	40m	75.8	86.1	42.2	40.7
S4	33m	75.6	85.8	44.9	31.4
S5	1h45m	82.1	89.6	63.5	55.6
S6	2h07m	82.5	89.9	**65.5**	57.3
S7	47m	79.3	87.9	57.7	49.8
S8	1h24m	81.0	88.9	58.2	53.5
S9	1h41m	82.1	89.6	65.1	**59.3**
S10	44m	74.0	85.2	32.7	27.7
S11	4h33m	82.7	89.8	64.3	57.9
S12	4h51m	82.6	89.9	64.7	58.7
S13	59m	78.0	87.4	49.7	44.8
S14	3h32m	82.0	89.5	60.0	55.6
S15	4h04m	**82.8**	**90.2**	63.5	59.1

TABLE 7.18: Multiclass SVM-based reordering model's performance. The reported time is in hours (h) and minutes (m).

In Section 6.2, we propose a dual multinomial logistic regression (Dual MLR) with a shrinking heuristic (Alrajeh and Niranjan, 2014b). We compare this shrinking approach with multiclass SVM in Figure 7.7.

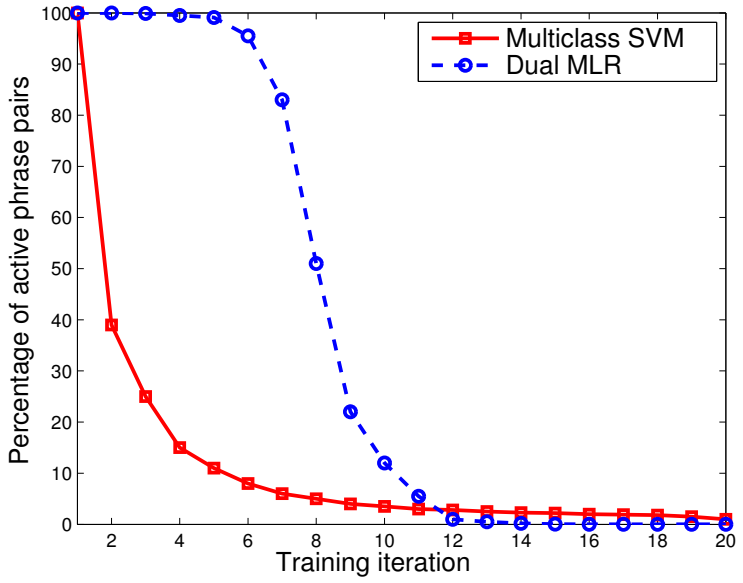

FIGURE 7.7: Comparison between multiclass SVM and dual multinomial logistic regression (MLR) in terms of active phrase pairs during training.

We also show training time and memory usage for each classifier when the number of phrase pairs increases in Figure 7.8 and Figure 7.9. The results show that multiclass

SVM consumes less memory than MaxEnt due to the shrinking technique discussed in Section 6.4.

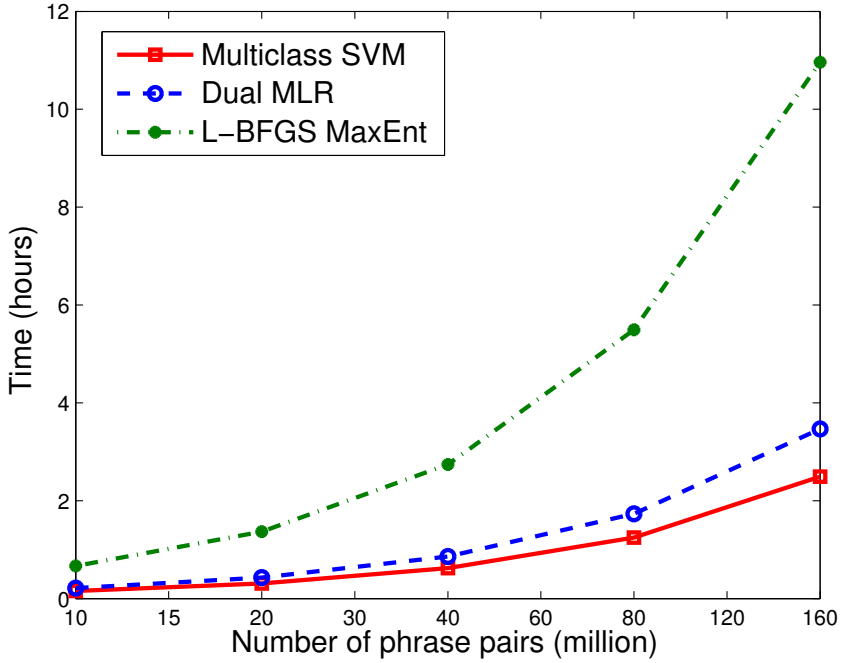

FIGURE 7.8: Training time for each classifier when the number of phrase pairs increases.

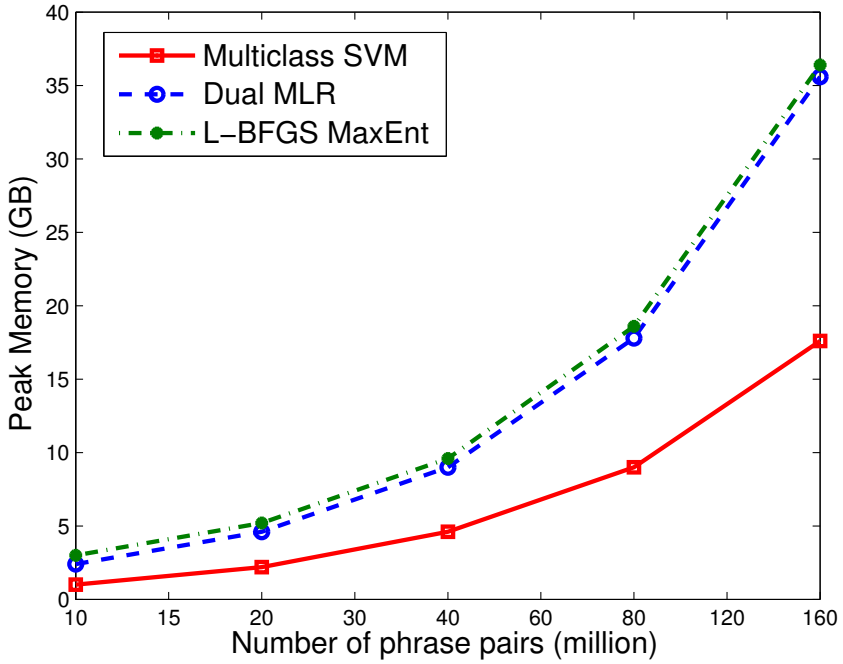

FIGURE 7.9: Memory usage of each classifier when the number of phrase pairs increases.

In Table 7.19, we compare MaxEnt and SVM with Ordinal Regression and Perceptron sub-models (Ni's approach). Ni (2010) simplified the perceptron-based learning problem to have as many sub-models as source phrases. Data were divided into small independent sets where samples having the same source phrase are considered a training set.

Reordering Model	Train Time	Accuracy	F_1 score
Maximum Entropy (S4)	1h50m	77.0	58.3
Maximum Entropy (S7)	3h10m	78.7	63.3
Multiclass SVM (S4)	33m	75.6	54.0
Multiclass SVM (S7)	47m	79.3	65.1
Ordinal Regression (S4)	1h14m	65.0	30.2
Ordinal Regression (S7)	2h48m	67.8	32.1
Perceptron sub-models (S4)	1h18m	79.9	66.3
Perceptron sub-models (S7)	1h5m	77.7	59.7

TABLE 7.19: Comparison of different discriminative reordering models.

Figure 7.10 illustrates a hypothetical problem to explain the main reason behind the low performance of Ordinal Regression. As shown, the model is restrictive due to the requirement of parallel discriminant hyperplanes.

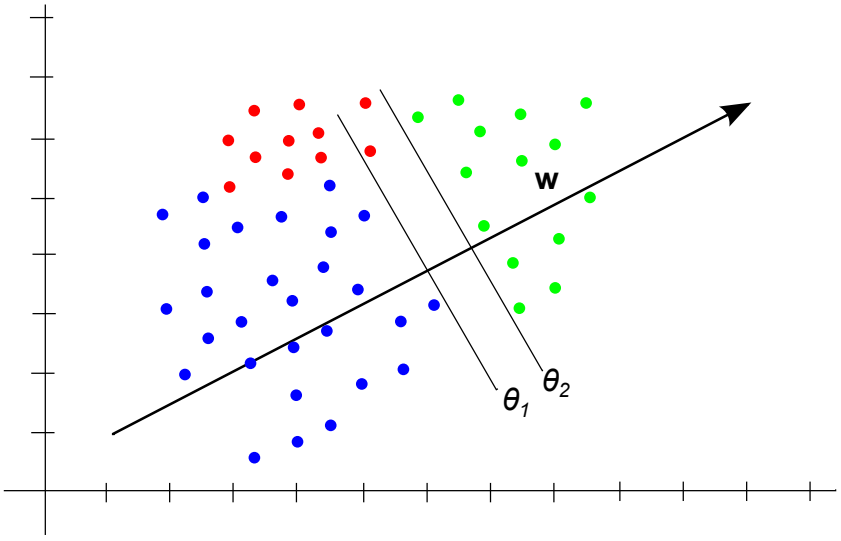

FIGURE 7.10: Ordinal Regression's hyperplanes for three unbalanced classes.

Finally, feature space can be reduced significantly by removing less frequent features. In our experiment, we only considered features occur more than five times in our corpus. Feature ranking is another method to select useful features for prediction (discussed in Section 4.2). Figure 7.11 shows the normalised mutual information for S7 features. The graph reveals that many features have low mutual information. Hence they are not related to the classification task and can be excluded from the model.

A ranking threshold for selecting features based on their mutual information is specified experimentally. In Figure 7.12, we tried different thresholds ranging from 0.005 to 0.05

and measured the accuracy of naive Bayes after each reduction. Note that the accuracy keeps increasing until 50% reduction. Then, the accuracy starts to decrease but the model maintains a good performance. After that, we see a dramatic loss. The results show that more than 70% of the feature space is not necessary. In fact, building the Bayesian model with full feature space reduced the accuracy.

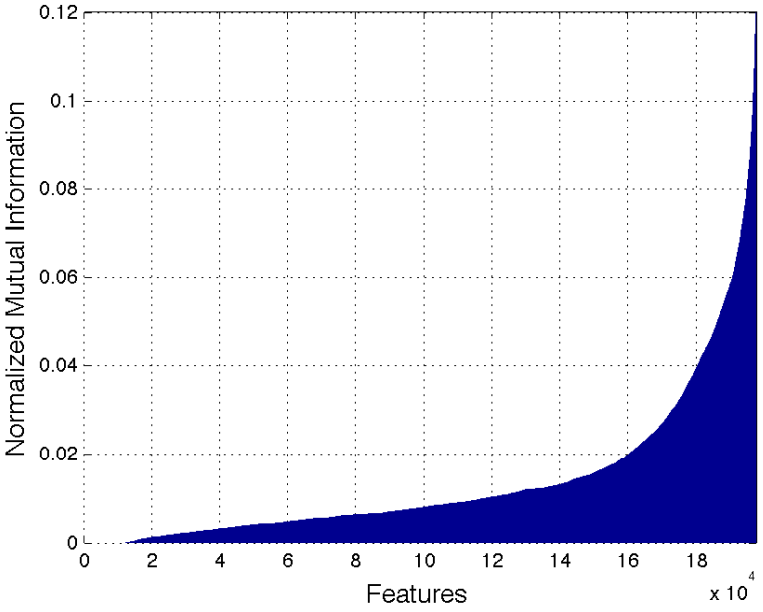

FIGURE 7.11: Normalised mutual information for S7 features (ranked from lowest to highest).

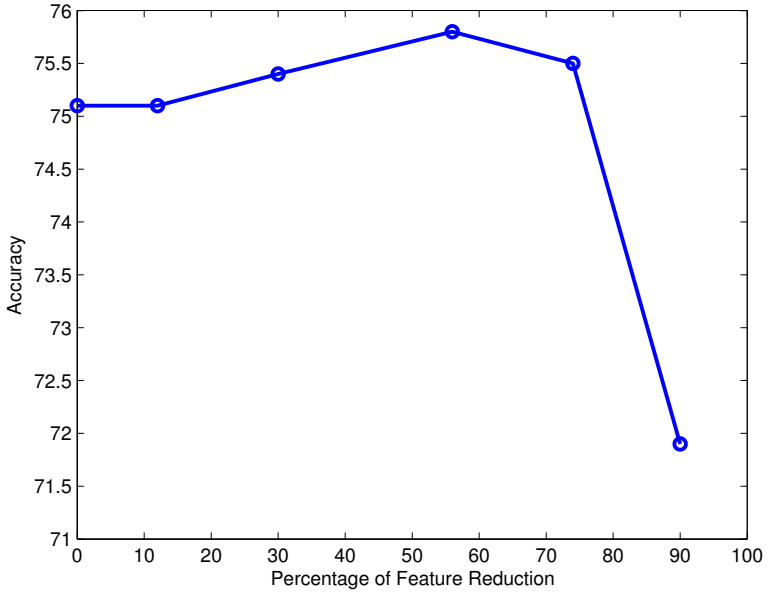

FIGURE 7.12: Classification accuracy of the Baysien model with different levels of feature reduction.

A more advanced feature selection method for discriminative models could be achieved by putting a Laplacian prior on models' parameters (i.e. L_1 regularisation). Laplacian prior forces irrelevant features to be zero as illustrated in Table 7.20.

Prior(**w**)	Number of Features	Feature Reduction	Disk Storage	Accuracy
Gaussian	1,635,154	–	82 MB	78.7 %
Laplacian	1,299,000	21 %	48 MB	79.1 %

TABLE 7.20: Impact of Gaussian and Laplacian priors on MaxEnt's model (S7).

7.2.2 Results of German to English

Figure 7.13 shows phrase pairs' distribution in our German-English corpus (Section 8.1).

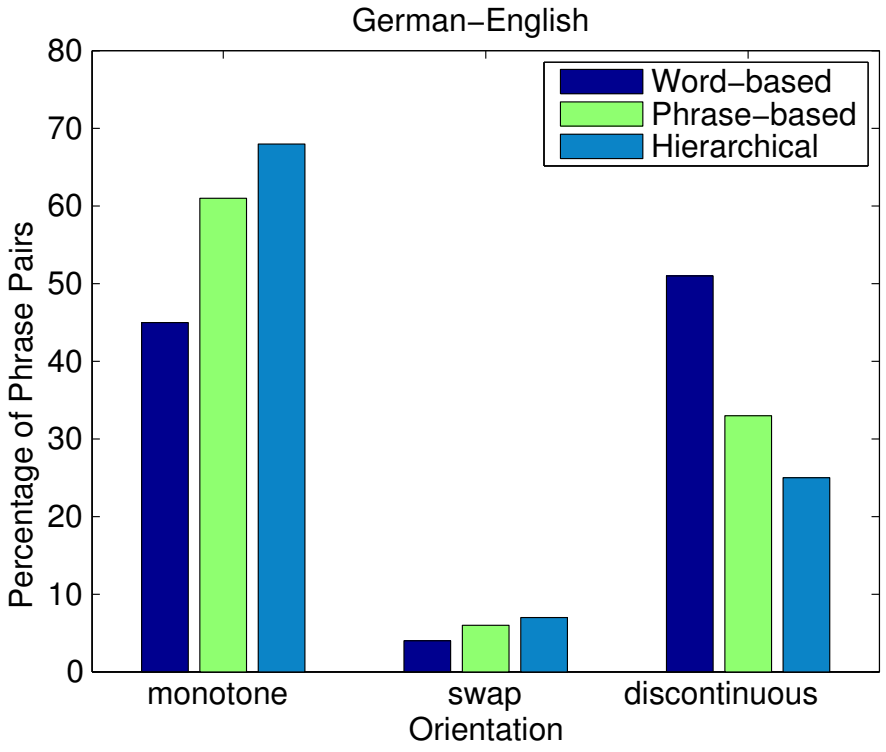

FIGURE 7.13: Distribution of German-English phrase pairs over three orientations estimated by word-based, phrase-based or hierarchical models.

Lexicalised reordering models' performance is presented in Table 7.21, 7.22 and 7.23. Note that the performance here is lower than what we have seen in the Arabic-English experiments. This indicates that the German-English reordering problem is harder.

Orientation	Confusion Matrix			Accuracy	F_1 score
	Monotone	Swap	Discontinuous	all classes	
Monotone	41.9	0.0	3.3		62.3
Swap	3.0	0.1	0.6	48.7	3.2
Discontinuous	44.4	0.0	6.8		21.9

TABLE 7.21: The performance of lexicalised reordering model (word-based).

Orientation	Confusion Matrix			Accuracy	F_1 score
	Monotone	Swap	Discontinuous	all classes	
Monotone	59.9	0.0	1.3		76.4
Swap	5.8	0.1	0.5	62.5	3.8
Discontinuous	29.9	0.1	2.5		13.5

TABLE 7.22: The performance of lexicalised reordering model (phrase-based).

Orientation	Confusion Matrix			Accuracy	F_1 score
	Monotone	Swap	Discontinuous	all classes	
Monotone	67.1	0.1	0.9		81.4
Swap	6.5	0.1	0.4	69.0	3.9
Discontinuous	23.1	0.1	1.7		12.5

TABLE 7.23: The performance of lexicalised reordering model (hierarchical).

As in the Arabic-English experiments, we compare naive Bayes, MaxEnt and multiclass SVM under various feature sets. The results are reported in Table 7.24, 7.25 and 7.26. Surprisingly, the performance of all our models is not far from lexicalised reordering models' performance.

Feature Set	Train Time	Accuracy	F_1 score		
			Monotone	Swap	Discontinuous
S1	37m	64.7	78.1	2.5	19.5
S2	33m	65.3	78.3	8.0	28.2
S3	38m	66.2	78.4	7.4	40.3
S4	43m	65.0	77.7	13.7	33.0
S5	2h21m	64.0	76.2	**27.7**	43.1
S6	2h52m	62.2	74.5	27.2	43.1
S7	49m	66.8	78.7	12.9	43.0
S8	1h31m	**68.7**	**80.3**	22.4	40.7
S9	1h51m	67.3	78.6	21.8	43.4
S10	58m	64.2	77.2	12.2	34.5
S13	6h4m	66.2	78.3	11.5	**43.7**

TABLE 7.24: Naive Bayes reordering model's performance.

Finally, we compare MaxEnt and SVM with Ordinal Regression and Perceptron sub-models (Ni's approach) as shown in Table 7.27.

Feature Set	Train Time	Accuracy	F_1 score		
			Monotone	Swap	Discontinuous
S1	2h34m	68.0	80.9	0.1	1.7
S2	1h58m	68.2	81.0	0.6	6.0
S3	2h46m	69.4	81.5	1.3	28.4
S4	3h5m	68.3	81.0	3.1	13.2
S5	7h21m	69.6	81.5	8.5	27.9
S6	9h42m	69.6	81.6	8.3	26.6
S7	3h46m	70.0	81.5	14.0	41.8
S8	6h38m	**71.3**	82.3	21.1	44.7
S9	7h11m	**71.3**	**82.4**	**23.2**	**45.6**
S10	3h37m	67.3	80.0	4.7	22.7
S13	4h15m	69.4	81.2	11.3	41.1

TABLE 7.25: Maximum entropy-based reordering model's performance.

Feature Set	Train Time	Accuracy	F_1 score		
			Monotone	Swap	Discontinuous
S1	42m	61.0	76.5	10.3	20.4
S2	38m	67.9	81.0	2.9	3.7
S3	49m	68.9	81.6	8.7	22.3
S4	47m	67.0	80.6	8.4	14.7
S5	2h27m	70.6	82.2	19.7	27.7
S6	3h0m	70.6	82.3	**20.2**	28.1
S7	58m	69.4	81.8	9.7	30.7
S8	1h52m	71.2	82.6	18.7	**34.8**
S9	2h11m	**71.4**	**82.7**	18.4	31.5
S10	1h3m	67.3	80.8	9.1	9.1
S13	1h19m	69.2	81.7	10.3	27.0

TABLE 7.26: Multiclass SVM-based reordering model's performance.

Reordering Model	Train Time	Accuracy	F_1 score
Maximum Entropy (S4)	3h5m	68.3	32.4
Maximum Entropy (S7)	3h46m	70.0	45.8
Multiclass SVM (S4)	47m	67.0	34.6
Multiclass SVM (S7)	58m	69.4	40.7
Ordinal Regression (S4)	2h31m	57.3	20.3
Ordinal Regression (S7)	3h4m	58.9	22.1
Perceptron sub-models (S4)	1h58m	69.3	40.8
Perceptron sub-models (S7)	1h19m	69.5	41.2

TABLE 7.27: Comparison of different discriminative reordering models.

7.3 Chapter Summary

We evaluated the classification models in two phases. First, the classifiers are examined on benchmark datasets in the field of pattern recognition in general. Second, reordering models are built based on these classifiers from an Arabic-English and German-English parallel corpora. Our multiclass SVM is shown to be superior to maximum entropy

classifier. It is a couple of times faster (nearly 4-fold) and more memory-efficient (50% reduction). The shrinking method in multiclass SVM is more effective than the one in dual MLR. The expanded space due to ANOVA mapping can be reduced significantly by removing less frequent features. In the next chapter, we examine the impact of our reordering models in translation systems. Several machine translation systems have been built, from Arabic to English and from German to English. The translation system are tested on well-known news translation benchmarks.

Chapter 8

Impact of Reordering Models on Translation Quality

In this chapter, we see reordering models' performance in a translation system. We present several machine translation systems from Arabic and German languages to English. The experiments were carried out on Moses - an open source system and the state-of-the-art in statistical machine translation (Koehn et al., 2007).

8.1 Corpora

The Arabic-English parallel corpus used in our experiments is a combination of MultiUN, ISI and Ummah to set up a large-scale corpus. MultiUN which is a large-scale parallel corpus extracted from the United Nations website[1] (Eisele and Chen, 2010). ISI and Ummah are on Linguistic Data Consortium[2] (LDC) with catalogue numbers (LDC2007T08) and (LDC2004T18), respectively. Table 8.2 shows general statistics of the corpora.

The ISI Arabic-English corpus was extracted automatically from two monolingual corpora: Arabic Gigaword Second Edition (LDC2006T02) and English Gigaword Second Edition (LDC2005T12). The data domain is news articles published by Xinhua News Agency and Agency France Press. The extraction was done using the automatic parallel sentence identification method described in Munteanu and Marcu (2005). The parallel sentence identification approach is designed to judge sentence pairs in isolation from their contexts, and can therefore find parallel sentences within document pairs which are not parallel. For each sentence pair in the corpus, the names of the documents from which the two sentences were extracted are provided, as well as a confidence score

[1] http://www.ods.un.org/ods/
[2] http://ldc.upenn.edu/

(between 0.5 and 1.0), which is an indicative of their degree of parallelism. In our experiments, we selected sentences with more than 0.9 confidence score. Table 8.1 shows the score distribution.

Confidence Score	Sentence Pairs [%]
0.9 - 1.0	81.0
0.8 - 0.9	7.0
0.7 - 0.8	4.8
0.6 - 0.7	3.8
0.5 - 0.6	3.4

TABLE 8.1: Confidence score of ISI Arabic-English corpus parallelism

Table 8.2 shows general statistics of the three corpora. There are two observations we can state. First, English sentences are usually longer than Arabic sentences. This leads to make the English side always larger. Second, Arabic language is richer than English because its vocabulary are doubled the English vocabulary. These observations are related to language richness because rich language express meaning in short sentences. Hence it requires diverse words.

Corpus	MultiUN		ISI		Ummah	
Statistics	Arabic	English	Arabic	English	Arabic	English
Sentence Pairs	9.7 M		1.1 M		80 K	
Running Words	255.5 M	285.7 M	30.5 M	34.4 M	2.7 M	2.9 M
Word/Line	22	25	27	31	33	36
Vocabulary Size	677 K	410 K	354 K	195 K	63 K	46 K
Vocabulary %	0.26%	0.14%	1.16%	0.57%	2.33%	1.59%

TABLE 8.2: General statistics of three Arabic-English corpora: MultiUN, ISI and Ummah (M: million, K: thousand).

The German-English parallel corpus used in our experiments is a combination of Europarl, Common Crawl and News Commentary. They are available on the WMT workshop website[3]. Table 8.3 shows general statistics of the corpora.

Corpus	Europarl		Common Crawl		News Commentary	
Statistics	German	English	German	English	German	English
Sentence Pairs	1.9 M		2.4 M		0.2 M	
Running Words	44.6 M	47.9 M	47. M	51.4 M	4.5 M	4.4 M
Word/Line	23	25	20	22	22	22
Vocabulary Size	649 K	305 K	273 K	172 K	250 K	155 K
Vocabulary %	1.46%	0.64%	0.58%	0.33%	5.56%	3.52%

TABLE 8.3: General statistics of three German-English corpora: Europarl, Common Crawl and News Commentary (M: million, K: thousand).

[3] http://www.statmt.org/wmt14/

8.2 Experimental Design

We used the Moses toolkit (Koehn et al., 2007) with its default settings. We set the distortion limit to be 10 and used the early distortion cost (Menezes and Quirk, 2005). The language model is a 5-gram built from the English side with interpolation and Kneser-Ney smoothing (Kneser and Ney, 1995). We tuned the system using PRO technique (Hopkins and May, 2011). We built several Arabic-English and German-English translation systems. First system does not have a reordering model only a distortion penalty. Second system has a hierarchical lexicalised reordering model by specifying the configuration string `hier-msd-backward-fe`. Sparse reordering features (Cherry, 2013) are included in the third system. We only used '`sparse-phrase=1`' option with top 200 words. The rest have our reordering models with different feature sets.

As commonly used in statistical machine translation, we evaluated the translation performance by BLEU score (Papineni et al., 2002) and NIST (Doddington, 2002). We also computed statistical significance for the proposed models using a *paired bootstrap resampling* method (Koehn, 2004b).

8.3 Results of Arabic to English

Test sets are from NIST MT06 and MT08 where the Arabic sides are 1797 and 813 sentences, respectively. Each sentence has four English references. Table 8.4 reports the size of each reordering model. Note that there is a big difference between the lexicalised model and the feature-based models.

Reordering Model	Parameters (million)	Disk Storage (GB)
Lexicalised	73.2	5.9
Maximum Entropy (S6)	20.0	0.8
Maximum Entropy (S7)	3.2	0.2
Multiclass SVM (S6)	17.1	0.7
Multiclass SVM (S7)	2.4	0.1
Naive Bayes (S5)	16.3	0.7
Naive Bayes (S7)	2.1	0.1
Perceptron sub-models (S4)	29.4	0.8
Perceptron sub-models (S7)	18.0	1.1

TABLE 8.4: Comparison of problem sizes in terms of number of parameters and storage for the different models (S6 and S7 are different feature set see Table 7.11 for more details).

Table 8.5 presents NIST and BLEU scores for eleven translation systems in MT06 and MT08 test sets. Our models achieve improvements on top of a strong baseline system with sparse reordering features. Note that feature sets (S6) and (S7) have similar scores although (S6) has higher classification accuracy in Table 7.18.

	MT06				MT08			
Phrase-based SMT	NIST	Δ	BLEU	Δ	NIST	Δ	BLEU	Δ
No Reordering Model	9.1	-0.3	35.5	-1.6	9.8	-0.3	41.0	-1.9
LexicalRM (**baseline**)	9.4	–	37.1	–	10.1	–	42.9	–
LexicalRM + sparseRM	9.5	+0.1	37.6	+0.5	10.3	+0.2	43.8	+0.9
MaxEnt-RM (S6) + sparseRM	9.6	+0.2	38.1	+1.0	10.4	+0.3	44.5	+1.6
MaxEnt-RM (S7) + sparseRM	9.6	+0.2	38.1	+1.0	10.4	+0.3	44.4	+1.5
SVM-RM (S6) + sparseRM	9.6	+0.2	38.1	+1.0	10.4	+0.3	44.4	+1.5
SVM-RM (S7) + sparseRM	9.6	+0.2	38.0	+0.9	10.4	+0.3	44.3	+1.4
NaiveBayes-RM (S5) + sparseRM	9.4	–	37.5	+0.4	10.3	+0.2	43.9	+1.0
NaiveBayes-RM (S7) + sparseRM	9.5	+0.1	37.8	+0.7	10.3	+0.2	44.0	+1.1
Percep-SubRMs (S4) + sparseRM	9.4	–	37.0	-0.1	10.1	–	42.7	-0.2
Percep-SubRMs (S7) + sparseRM	9.4	–	37.4	+0.3	10.2	+0.1	43.2	+0.3

TABLE 8.5: Arabic-English Translation results for two evaluation sets measured with BLEU [%] and NIST (RM: Reordering Model)

	NewsTest2013				NewsTest2014			
Phrase-based SMT	NIST	Δ	BLEU	Δ	NIST	Δ	BLEU	Δ
No Reordering Model	7.4	-0.2	26.5	-0.9	7.5	-0.2	26.7	-1.0
LexicalRM (**baseline**)	7.6	–	27.4	–	7.7	–	27.7	–
LexicalRM + sparseRM	7.6	–	27.5	+0.1	7.8	+0.1	28.0	+0.3
MaxEnt-RM (S6) + sparseRM	7.7	+0.1	27.8	+0.4	7.9	+0.2	28.2	+0.5
MaxEnt-RM (S7) + sparseRM	7.7	+0.1	27.7	+0.3	7.9	+0.2	28.1	+0.4
SVM-RM (S6) + sparseRM	7.7	+0.1	27.8	+0.4	7.9	+0.2	28.1	+0.4
SVM-RM (S7) + sparseRM	7.7	+0.1	27.7	+0.3	7.9	+0.2	28.1	+0.4
NaiveBayes-RM (S5) + sparseRM	7.6	–	27.5	+0.1	7.8	+0.1	28.0	+0.3
NaiveBayes-RM (S7) + sparseRM	7.6	–	27.5	+0.1	7.8	+0.1	28.0	+0.3
Percep-SubRMs (S4) + sparseRM	7.6	–	27.3	-0.1	7.7	–	27.6	-0.1
Percep-SubRMs (S7) + sparseRM	7.6	–	27.4	–	7.7	–	27.8	+0.1

TABLE 8.6: German-English Translation results for two evaluation sets measured with BLEU [%] and NIST (RM: Reordering Model)

8.4 Results of German to English

For German-English systems, we used compound splitting (Koehn and Knight, 2003). The language model in these systems is built from both the English side and News Crawl (2007-1013) which is available on the WMT workshop website[4]. Both development and test sets are also taken from the WMT workshop website. We have chosen news from 2010 as development set and news from 2013 and 2014 as test sets. The English sides are 43,455 words (2015 sentences) and 56,089 words (3000 sentences), respectively. Table 8.6

[4] http://www.statmt.org/wmt14/

presents NIST and BLEU scores for eleven translation systems in newstest2013 and newstest2014.

8.5 Chapter Summary

Posing phrase movements as a classification problem, we exploit recent developments in solving large-scale multiclass support vector machines using stochastic gradient learning algorithm and show significant advantages in Arabic-English and German-English systems. The algorithms we propose are shown to be computationally fast and memory-efficient. In terms of evaluating translation quality using the BLEU score, we achieve 1.0 point in MT06 and 1.6 in MT08 over a lexicalised reordering model with at least 95% statistical significance in Arabic-English systems. We also achieve comparable improvements in German-English systems. Interestingly, We found that a reordering model based on alignments features (S7) is much more compact than using boundaries features (S6) without reduction in translation quality.

Chapter 9

Conclusions

In machine translation, variation in grammatical structures between source and target languages can cause large movements of phrases (e.g. Arabic-English). Modelling such movements in phrase-based SMT systems is crucial in achieving translations of long sentences that appear natural in the target language. This dissertation poses the phrase reordering problem as a classification problem and solves it by machine learning methods; this requires working with very large corpora so that the context dependent long range phrase movements may be captured by a learning algorithm. A large-scale parallel corpus (e.g. hundred million words) is important for improving such reordering model, this improvement comes at a price of computational complexity. It is particularly pronounced when discriminative models are considered such as maximum entropy-based model. In this thesis, we propose a number of techniques able to reach solutions on top-end desktop machines while competing methods cannot.

Modelling phrase reordering involves three parts: formulating phrase movements, phrase linguistic features and learning agent. In the first part, we review three approaches to define a phrase orientation (Chapter 3): word-based (Ni, 2010), phrase-based (Tillmann, 2004) and hierarchical (Galley and Manning, 2008). In this thesis, we adopt the third approach because it is able to tackle long-distance reorderings by taking into account the hierarchical structure of a sentence. In the second part, we have seen several feature extraction methods (Zens and Ney, 2006; Xiong et al., 2006; Nguyen et al., 2009; Ni et al., 2011) and propose novel features namely word alignment features able to capture more reordering evidences (Chapter 4). Interestingly, we found that a reordering model based on alignments features is more compact (80% reduction) than using boundaries features (S6) without losing translation quality. In the last part, we explore several generative and discriminative learning approaches detailed below (Chapter 5 and 6).

We propose Bayesian naive Bayes to model phrase movements. Our Bayesian model using a Dirichlet prior is shown to be superior to the lexicalised model of estimating probabilities as relative frequencies of phrase movements. The training time of naive

Bayes is as fast as the lexicalised model and its storage requirement is more than 10 times smaller. Discriminative models might achieve higher score than naive Bayes. However, its parameter estimation requires only one pass over the data with limited memory (i.e. no iterative learning). This is a critical advantage over discriminative models particularly for very large corpora. For example, the training time of MaxEnt model on a large Arabic-English corpus (more than 250 million words) is fifteen hours while our Bayesian model required only three and half hours.

In discriminative learning, we propose computationally fast and memory-efficient algorithms to learn SVM, Multiclass SVM and Multinomial Logistic Regression. Using dual coordinate descent methods for learning, we provide a mechanism to shrink the amount of training data required for each iteration. Hence, we produce significant saving time (4-fold speed) and memory (50% reduction) while preserving the accuracy of the models. These efficient classifiers allow us to build large-scale discriminative reordering models. Experiments were carried out on Arabic-English and German-English corpora with more than a quarter of a billion words. In terms of the BLEU score, we achieve 1.0 point in MT06 and 1.5 in MT08 over a lexicalised reordering model with at least 95% statistical significance in Arabic-English systems. We also achieve comparable improvements in German-English systems.

We empirically show that the approach of Ni et al. (2011) to break down the learning complexity into small sub-models is not necessary. In fact, having one reordering model is more beneficial to a machine translation system. Although the number of parameters for each sub-model is small, the overall parameters are larger than having one model incorporates all the training data.

Finally, a classification model with flexible margins between classes (Ni et al., 2010b) is a step towards respecting phrase reordering structure. Ni et al. (2011) suggest to take the idea further and formulates phrase reordering problem as ordinal regression rather than classification. In our experiments, the accuracy of ordinal regression is worse than the lexicalised model (baseline). In Arabic-English experiments, for example, ordinal regression has 67.8% accuracy while the lexicalised model is 8 points higher (75.9%). We found that ordinal regression is restrictive due to the requirement of parallel discriminant hyperplanes. This results in a low accuracy compared to SVM and Multinomial Logistic Regression. Another learning approach is Voted Spheres which we conclude to be not suited for our problem. It has been introduced by Farran and Saunders (2009) to tackle large-scale classification. Its non-parametric nature is attractive. We found that the classifier has high accuracy. However, it is very slow during prediction phase (i.e. more 100 times slower than competing classifiers) which makes it not practical for machine translation systems.

9.1 Future Work

The work in this thesis revealed several interesting areas for future work. We summarise them in four points as follows:

- Many problems in SMT including phrase reordering can be formulated as a structured prediction problem. Taking the whole sentence into account when predicting phrase reorderings is challenging. In a multiclass setting, the number of possible permutations is large and difficult to train. This problem can be solved by assuming that the output is structured. Algorithms proposed for sequence tagging can be adopted such as structured perceptron (Collins, 2002). Ni (2010), in his thesis, theoretically discuss structured prediction technologies for machine translation problems but conclude that in practice the approach might be infeasible even to a medium-scale corpus. Instead of that, he borrow the idea of flexible margins between classes in structured learning and apply a perceptron-based algorithm. This variable-margin classification approach significantly reduces the computational complexity. We believe phrase reordering as a parsing problem still can be applied to a large-scale corpus. Feng et al. (2012) propose a reordering model based on sequence labelling techniques. For a sentence pair, they assign each source word an orientation label. Then they let an algorithm such as CRFs or RNN to do the learning task. The computational costs are high (around 16 hours and 120G RAM). Further research is needed to reduce these costs particularly the memory consumption. Furthermore, a potential advancement might be achieved by learning phrase's label rather than word's label.

- Most of the work in this thesis is to build large-scale reordering models for phrase-based translation systems. These models are usually not considered in hierarchical or tree-based systems since reordering is modelled implicitly by the grammar rules. Recently, He et al. (2010) classify the grammar rules into reordering patterns and build context-based MaxEnt models to select the right pattern during decoding. Similarly Huck et al. (2012) build one MaxEnt model instead of different models for each reordering pattern. We think that the methods proposed in this thesis are applicable to the hierarchical systems.

- Cherry (2013) proposed using sparse features to optimise BLEU with the decoder instead of training a classifier independently. The reported results shows that sparse decoder features are superior to maximum entropy classifier although their training sets is limited to a few thousand sentences. Recently, Auli et al. (2014) train a discriminative reordering model with millions of sparse features on a large-scale corpus. They demonstrate that the expected BLEU objective is more effective than likelihood training. This interesting direction deserves more efforts.

- All the previous experiments were for improving the reordering model. Analogous to it, translation model can be formulated as a classification problem. In reordering model, the phrase movements are considered classes while in translation model phrase translations are the classes. For example, Ni et al. (2010b) use max-margin structure (MMS) model to predict translation probabilities.

Appendix A

Mathematical Derivations

A.1 MAP Estimate for Naive Bayes

Multinomial distribution is defined as:

$$p(\mathbf{x}|\mathbf{q}) = C \prod_m q_m^{x_m} \tag{A.1}$$

where C is a multinomial coefficient,

$$C = \frac{(\sum_m x_m)!}{\prod_m x_m!}, \tag{A.2}$$

and q_m is an event probability ($\sum_m q_m = 1$).

A maximum a posteriori probability (MAP) estimate requires a prior over $\mathbf{q}$. Dirichlet distribution is a conjugate prior and is defined as:

$$p(\mathbf{q}|\boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_m \alpha_m\right)}{\prod_m \Gamma(\alpha_m)} \prod_m q_m^{\alpha_m - 1} \tag{A.3}$$

where α_m is a is a parameter with a positive value.

Finding the MAP estimate for $\mathbf{q}$ given a data is as follows:

$$\begin{aligned}
\mathbf{q}^* &= \underset{\mathbf{q}}{\text{argmax}}\ p(\mathbf{q}|\boldsymbol{\alpha}, \mathbf{X}) \\
&= \underset{\mathbf{q}}{\text{argmax}}\ \{p(\mathbf{q}|\boldsymbol{\alpha})p(\mathbf{X}|\mathbf{q})\} \\
&= \underset{\mathbf{q}}{\text{argmax}}\ \left\{p(\mathbf{q}|\boldsymbol{\alpha}) \prod_n p(\mathbf{x}_n|\mathbf{q})\right\} \\
&= \underset{\mathbf{q}}{\text{argmax}}\ \left\{\prod_m q_m^{\alpha_m - 1} \prod_{n,m} q_m^{x_{nm}}\right\} \\
&= \underset{\mathbf{q}}{\text{argmax}}\ \left\{\sum_m \log q_m^{\alpha_m - 1} + \sum_{n,m} \log q_m^{x_{nm}}\right\}.
\end{aligned} \tag{A.4}$$

Since our function is subject to constraints ($\sum_m q_m = 1$), we introduce Lagrange multiplier as follows:

$$f(\mathbf{q}) = \sum_m \log q_m^{\alpha_m - 1} + \sum_{n,m} \log q_m^{x_{nm}} - \lambda(\sum_m q_m - 1). \tag{A.5}$$

Now we can find $\mathbf{q}^*$ by taking the partial derivative with respect to one variable q_m:

$$\begin{aligned}
\frac{\partial f(\mathbf{q})}{\partial q_m} &= \frac{\alpha_m - 1 + \sum_n x_{nm}}{q_m} - \lambda \\
q_m &= \frac{\alpha_m - 1 + \sum_n x_{nm}}{\lambda}.
\end{aligned} \tag{A.6}$$

Finally, we sum both sides over M to find λ :

$$\begin{aligned}
\lambda \sum_m q_m &= \sum_m \left(\alpha_m - 1 + \sum_n x_{nm}\right) \\
\lambda &= \sum_m (\alpha_m - 1) + \sum_{n,m} x_{nm}.
\end{aligned} \tag{A.7}$$

The solution can be simplified by choosing the same value for each α_m which will result in:

$$q_m = \frac{\alpha - 1 + \sum_n x_{nm}}{M(\alpha - 1) + \sum_{n,m'} x_{nm'}}. \tag{A.8}$$

A.2 Bayesian Inference for Naive Bayes

In Appendix A.1, the inference is based on a single point estimate of $\mathbf{q}$ that has the highest posterior probability. However, it can be based on the whole parameter space to incorporate uncertainty. The probability of a new data point marginalized over the posterior as follows:

$$p(\mathbf{x}|\boldsymbol{\alpha}, \mathbf{X}) = \int p(\mathbf{x}|\mathbf{q})p(\mathbf{q}|\boldsymbol{\alpha}, \mathbf{X})\,\mathrm{d}\mathbf{q}, \tag{A.9}$$

$$p(\mathbf{q}|\boldsymbol{\alpha}, \mathbf{X}) = \frac{p(\mathbf{q}|\boldsymbol{\alpha})p(\mathbf{X}|\mathbf{q})}{\int p(\mathbf{q}|\boldsymbol{\alpha})p(\mathbf{X}|\mathbf{q})\mathrm{d}\mathbf{q}}. \tag{A.10}$$

Since Dirichlet and Multinomial distributions are conjugate pairs, they form the same density as the prior. Therefore the posterior is also Dirichlet. Now we can expand the posterior expression and re-arrange it to look like a Dirichlet as follows:

$$\begin{aligned} p(\mathbf{q}|\boldsymbol{\alpha}, \mathbf{X}) &\propto p(\mathbf{q}|\boldsymbol{\alpha}) \prod_n p(\mathbf{x}_n|\mathbf{q}) \\ &\propto \prod_m q_m^{\alpha_m - 1} \prod_n \prod_m q_m^{x_{nm}} \\ &\propto \prod_m q_m^{(\alpha_m + \sum_n x_{nm}) - 1}. \end{aligned} \tag{A.11}$$

The new hyperparameters of the posterior is:

$$\alpha_m^* = \alpha_m + \sum_n x_{nm}. \tag{A.12}$$

Finally, we expand and re-arrange Dirichlet and multinomial distributions inside the integral in (A.9) as follows:

$$\begin{aligned} &p(\mathbf{x}|\boldsymbol{\alpha}, \mathbf{X}) = \\ &\int C \prod_m q_m^{x_m} \frac{\Gamma\left(\sum_m \alpha_m^*\right)}{\prod_m \Gamma(\alpha_m^*)} \prod_m q_m^{\alpha_m^* - 1}\,\mathrm{d}\mathbf{q} \\ =&C \frac{\Gamma\left(\sum_m \alpha_m^*\right)}{\prod_m \Gamma(\alpha_m^*)} \int \prod_m q_m^{\alpha_m^* + x_m - 1}\,\mathrm{d}\mathbf{q}. \end{aligned} \tag{A.13}$$

Note that inside the integral looks a Dirichlet without a normalizing constant. If we multiply and divide by its normalizing constant (i.e. Beta function), the integral is going to be one because it is a density function, resulting in:

$$
\begin{aligned}
& p(\mathbf{x}|\boldsymbol{\alpha}, \mathbf{X}) = C \, \frac{\Gamma\left(\sum_m \alpha_m^*\right)}{\prod_m \Gamma(\alpha_m^*)} \\
& \mathrm{B}(\boldsymbol{\alpha}^* + \mathbf{x}) \int \frac{1}{\mathrm{B}(\boldsymbol{\alpha}^* + \mathbf{x})} \prod_m q_m^{\alpha_m^* + x_m - 1} \, \mathrm{d}\mathbf{q}_c \\
= & C \, \frac{\Gamma\left(\sum_m \alpha_m^*\right)}{\prod_m \Gamma(\alpha_m^*)} \, \mathrm{B}(\boldsymbol{\alpha}^* + \mathbf{x}) \\
= & C \, \frac{\Gamma\left(\sum_m \alpha_m^*\right)}{\prod_m \Gamma(\alpha_m^*)} \, \frac{\prod_m \Gamma(\alpha_m^* + x_m)}{\Gamma\left(\sum_m (\alpha_m^* + x_m)\right)}.
\end{aligned}
\tag{A.14}
$$

Appendix B

Matlab Scripts

B.1 Multiclass Perceptron

Matlab code of multiclass perceptron described in section 5.3:

```
function confusion = perceptron(Xtr, ytr, Xts, yts)

%% perceptron(Xtr, ytr, Xts, yts)
% Multi-class Perceptron with loss function between neighbour classes
% Return a confusion matrix between true and predicted classes
% Xtr and ytr are training data
% Xts and yts are testing data

%% Initisliase the parameters
C=4; % Number of classes
[N,M] = size(Xtr); % N examples in M dimensions
W = zeros(M,C); % Start at zero
tol = 0.001; % Stopping tolerance
Nits = 100; % Maximum number of iterations
delta=abs(ytr*ones(1,C)-ones(N,1)*(1:C)); % loss function
id=eye(C);
T=id(ytr,:); % Convert to binary
change = inf;
it = 0;
step=0.1;

%% Learning W
while change>tol & it<=Nits
    % Prediction
    P=Xtr*W+delta;
    P=(P==max(P,[],2)*ones(1,C));

    % Gradient
    grad = Xtr'*(T-P);

    % Update W
    W_pre = W;
    W = W + step*grad;

    it = it + 1;
```

```
    change = sum(sum((W - W_pre).^2));
end

%% Confusion matrix
% Assign to max probability
[foo c] = max(Xts*W,[],2);
confusion = zeros(C);
for predicted = 1:C
    for true = 1:C
        confusion(true,predicted) = ...
            confusion(true,predicted) + sum(yts==true & c==predicted);
    end
end
```

B.2 Naive Bayes

Matlab code of naive Bayes described in section 6.1:

```
function confusion = naivebayes(Xtr, ytr, Xts, yts)

%% naivebayes(Xtr, ytr, Xts, yts)
% Naive Bayes with Bayesian Inference (Multinomial-Dirichlet)
% Return a confusion matrix between true and predicted classes
% Xtr and ytr are training data
% Xts and yts are testing data

%% Initisliase the parameters
C=4; % Number of classes
Ntr = size(Xtr,1); % N training examples
Nts = size(Xts,1); % N testing examples
alpha = 2; % Dirichlet prior parameter

%% Compute the test probabilities
% Do this with logs for numerical stability.
prior=zeros(1,C);
logP = zeros(Nts,C);
for c = 1:C
    % Prior class probabilities
    prior(1,c) = sum(ytr==c)/Ntr;

    % Class conditional probabilities
    alpha_c = alpha+sum(Xtr(ytr==c,:),1);
    Xalpha_c = Xts+ones(Nts,1)*alpha_c;
    logP(:,c) =  sum(log(1:(sum(alpha_c)-1))) - sum(log(gamma(alpha_c)))...
              +  sum(log(gamma(Xalpha_c)),2) - gammaln(sum(Xalpha_c,2));
end

%% Normalise
logP = logP + ones(Nts,1)*log(prior);
P = exp(logP);
P = P./(sum(P,2)*ones(1,C));

%% Confusion matrix
% Assign to max probability
[foo c] = max(P,[],2);
```

```
confusion = zeros(C);
for predicted = 1:C
    for true = 1:C
        confusion(true,predicted) = ...
            confusion(true,predicted) + sum(yts==true & c==predicted);
    end
end
```

B.3 Multinomial Logistic Regression

Matlab code of multinomial logistic regression described in section 5.4:

```
function confusion = softreg(Xtr, ytr, Xts, yts)

%% softreg(Xtr, ytr, Xts, yts)
% Softmax Regression with maximum a posteriori (MAP) estimate
% Return a confusion matrix between true and predicted classes
% Xtr and ytr are training data
% Xts and yts are testing data

%% Initisliase the parameters
C=4; % Number of classes
[N,M] = size(Xtr); % N examples in M dimensions
W = zeros(M,C); % Start at zero
tol = 0.001; % Stopping tolerance
Nits = 100; % Maximum number of iterations
ss = 10; % Prior variance on the parameters of w
id=eye(C);
T=id(ytr,:); % Convert to binary
change = inf;
it = 0;
step=0.1;

%% Learning W
while change>tol & it<=Nits
    % Prediction
    E = exp(Xtr*W);
    P = E./(sum(E,2)*ones(1,C));

    % Gradient
    grad = -(1/ss)*W + Xtr'*(T-P);

    % Update W
    W_pre = W;
    W = W + step*grad;

    it = it + 1;
    change = sum(sum((W - W_pre).^2));
end

%% Confusion matrix
% Assign to max probability
E = exp(Xts*W);
P = E./(sum(E,2)*ones(1,C));
```

```
[foo , c] = max(P,[],2);

confusion = zeros(C);
for predicted = 1:C
    for true = 1:C
        confusion(true,predicted) = ...
            confusion(true,predicted) + sum(yts==true & c==predicted);
    end
end
```

B.4 Data Preparation

Matlab code to prepare Data for IBM models:

```
function [corpus_f corpus_e] = prepareData(source, target)

%% prepareData(source, target)
% Given parallel text files, return two structures have
% source and target data

%% Read source file
fid = fopen(source); % French corpus
C = textscan(fid, '%s','delimiter','\n'); % Read sentences from text file
corpus_f.sen = regexp(C{1,1},'\w*','match'); % Tokenize each sentence
C = cellfun(@(x) {char(x)},corpus_f.sen);
corpus_f.voc = unique(lower(cellstr(char(C{:})))); % Vocabulary
corpus_f.vocSize = size(corpus_f.voc,1); % Vocabulary size
fclose(fid);

%% Read target file
fid = fopen(target); % English corpus
C = textscan(fid, '%s','delimiter','\n');
corpus_e.sen = regexp(C{1,1},'\w*','match');
C = cellfun(@(x) {char(x)},corpus_e.sen);
corpus_e.voc = unique(lower(cellstr(char(C{:}))));
corpus_e.vocSize = size(corpus_e.voc,1);
fclose(fid);

% add empty word to the English Vocabulary which we assume to occur in each
% English sentence
corpus_e.vocSize = corpus_e.vocSize +1;
corpus_e.voc{corpus_e.vocSize} = 'NULL';

%% Text to number
% Converting French and English sentences into a numberized format in terms
% of their vocabularies
N = size(corpus_f.sen,1); % Number of parallel sentences
f = cell(N,1);
e = cell(N,1);

for s=1:N,
    s_len = size(corpus_f.sen{s},2); % length of each French sentence
    for w=1:s_len,
        f{s} = [f{s} find(strcmpi(corpus_f.voc,corpus_f.sen{s}{w}))];
    end
```

```
    s_len = size(corpus_e.sen{s},2); % length of each English sentence
    e{s} = corpus_e.vocSize;
    for w=1:s_len,
       e{s} = [e{s} find(strcmpi(corpus_e.voc,corpus_e.sen{s}{w}))];
    end
end

corpus_f.num = f;
corpus_e.num = e;
```

B.5 IBM model 1

Matlab code of IBM model 1 described in section 2.3.1.1:

```
function [transProb PP] = IBM_model1(corpus_f, corpus_e)

%% IBM_model1(corpus_f, corpus_e)
% Return translation probabilities, perplexity and entropy
% corpus_f and corpus_e are structures have source and target data,
% generated by prepareData function

%% Initisliase the parameters
N = size(corpus_f.sen,1); % Number of parallel sentences
f = corpus_f.num; % French sentences in a numberized format
e = corpus_e.num; % English sentences in a numberized format
Nits = 5; % Maximum number of iterations

% initializing translation probability of a French word given
% an English word, p(fn|em)
transProb=repmat(1/corpus_f.vocSize,corpus_f.vocSize,corpus_e.vocSize);

% sum of translation probabilities of a French word given any English word
% in the sentence, sum p(fj|ei) for all i
s_total=zeros(corpus_e.vocSize,1);

%% EM algorithm
for it=1:Nits
    % the expected number of times that an English word connects to a French
    % word in the translation (f|e), count(fn|em;f,e)
    count=zeros(corpus_f.vocSize,corpus_e.vocSize);

    %% Expectation step over all sentences
    for s=1:N,
        s_total(f{s})=sum(transProb(f{s},e{s}),2);
        % Evaluate the posterior, p(aj|ei,fj)=p(fj|ej)/s_total(fj)
        p_aj = transProb(f{s},e{s})./repmat(s_total(f{s}),1,size(e{s},2));

        % Collect fractional counts to yield the expectation
        % E{count(fn|em)} = sum_j sum_i p(aj|ei,fj)(fn==fj & em==ei)
        count(f{s},e{s})= count(f{s},e{s}) + p_aj;
    end

    %% Maximization step
    transProb = count./repmat(sum(count),corpus_f.vocSize,1);
```

```
end

%% perplexity
% perplexity is defined as 2 raised to the power of entropy, 2^H
% H(f|e) = - sum_s{p(f_s,e_s)log2 p(f_s/e_s)} = -1/N sum_s{log2 p(f_s/e_s)}
% H(f|e) = - logLikelihood/N
logLikelihood=0;
for s=1:N,
    % probability of an English word's position given a French word's position
    % and English sentence's length, p(aj=i|j,I)
    alignProb = 1/size(e{s},2);
    % translation probability of a French given an English sentence, p(f|e)
    logLikelihood = logLikelihood + ...
        sum( log2(alignProb) + log2(sum(transProb(f{s},e{s}),2)) );
end
PP = 2^(-logLikelihood/N);
```

B.6 IBM model 2

Matlab code of IBM model 2 described in section 2.3.1.1:

```
function [transProb alignProb PP] = IBM_model2(corpus_f, corpus_e, transProb)

%% IBM_model2(corpus_f, corpus_e)
% Return translation and alignment probabilities,  perplexity and entropy
% corpus_f and corpus_e are structures have source and target data,
% generated by prepareData function
% transProb from IBM model 1

%% Initisliase the parameters

if nargin < 3
    % transProb is not given
    % initializing translation probability of a French word given
    % an English word, p(fn|em)
    transProb=repmat(1/corpus_f.vocSize,corpus_f.vocSize,corpus_e.vocSize);
end

N = size(corpus_f.sen,1); % Number of parallel sentences
f = corpus_f.num; % French sentences in a numberized format
e = corpus_e.num; % English sentences in a numberized format
Nits = 5; % Maximum number of iterations

% sum of translation probabilities of a French word given any English word
% in the sentence, sum p(fj|ei) for all i
s_total=zeros(corpus_e.vocSize,1);

% alignment probability of position i given position j, p(aj=i|j,J,I;f,e)
A=zeros(100); % Maximum length of a sentence is 100
Asize=0;
alignProb=cell(1);
for s=1:N,
    J=size(f{s},2);
    I=size(e{s},2);
    if (A(J,I)==0),
```

```
        Asize = Asize+1;
        A(J,I)=Asize;
        alignProb(Asize)={ones(J,I)/I};
    end
end
count_a=cell(Asize,1);

%% EM algorithm
for it=1:Nits
    % the expected number of times that an English word connects to a French
    % word in the translation (f|e), count_t(fn|em;f,e)
    count_t=zeros(corpus_f.vocSize,corpus_e.vocSize);

    % the expected number of times that the word in position j of f is connected
    % to the word in position i of e, count_a(aj=i|j,J,I;f,e)
    for a=1:Asize,
        count_a(a)={0*alignProb{a}};
    end

    %%  Expectation step over all sentences
    for s=1:N,
        J=size(f{s},2);
        I=size(e{s},2);
        ta = transProb(f{s},e{s}).*alignProb{A(J,I)};
        s_total(f{s})=sum(ta,2);
        % Evaluate the posterior, p(aj|ei,fj)=p(fj|ej)p(aj)/s_total(fj)
        p_aj = ta./repmat(s_total(f{s}),1,I);

        % Collect fractional counts to yield the expectation
        % E{count_t(fn|em)} = sum_j sum_i p(aj|ei,fj)(fn==fj & em==ei)
        % E{count_t(aj=i|j,J,I)} = sum_j sum_i p(aj|ei,fj)
        count_t(f{s},e{s}) = count_t(f{s},e{s}) + p_aj;
        count_a(A(J,I)) = {count_a{A(J,I)} + p_aj};
    end

    %% Maximization step
    transProb = count_t./repmat(sum(count_t),corpus_f.vocSize,1);
    for a=1:Asize,
        alignProb(a) = {count_a{a}./repmat(sum(count_a{a},2),1,size(count_a{a},2))};
    end
end

%% perplexity
% perplexity is defined as 2 raised to the power of entropy, 2^H
% H(f|e) = - sum_s{p(f_s,e_s)log2 p(f_s/e_s)} = -1/N sum_s{log2 p(f_s/e_s)}
% H(f|e) = - logLikelihood/N
logLikelihood = 0;
for s=1:N,
    J=size(f{s},2);
    I=size(e{s},2);
    logLikelihood = logLikelihood + ...
        sum(log2(sum(transProb(f{s},e{s}).*alignProb{A(J,I)},2)));
end
PP = 2^(-logLikelihood/N);
```

Bibliography

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. Statistical machine translation. Technical report, Johns Hopkins University, Summer Workshop, 1999.

Abdullah Alrajeh and Mahesan Niranjan. Bayesian reordering model with feature selection. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 477–485, Baltimore, Maryland, USA, June 2014a. Association for Computational Linguistics.

Abdullah Alrajeh and Mahesan Niranjan. Large-scale reordering model for statistical machine translation using dual multinomial logistic regression. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1758–1763, Doha, Qatar, October 2014b. Association for Computational Linguistics.

Abdullah Alrajeh and Mahesan Niranjan. Generative and discriminative reordering models for statistical machine translation. In *Proceedings of the Eighth Saudi Students Conference*, London, United Kingdom, 2015a. Imperial College Press.

Abdullah Alrajeh and Mahesan Niranjan. Scalable reordering models for SMT based on multiclass SVM. *The Prague Bulletin of Mathematical Linguistics*, 103:65–84, 2015b.

Abdullah Alrajeh, Akiko Takeda, and Mahesan Niranjan. Memory-efficient large-scale linear support vector machine. In *Proceedings of SPIE: Seventh International Conference on Machine Vision (ICMV 2014)*, volume 9445, pages 944527–944527–6, Milan, Italy, February 2015. SPIE.

Galen Andrew and Jianfeng Gao. Scalable training of l1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 33–40. ACM, 2007. ISBN 978-1-59593-793-3.

Abhishek Arun and Philipp Koehn. Online learning methods for discriminative training of phrase based statistical machine translation. In *Proceedings of the MT Summit XI*, Copenhagen, Denmark, September 2007. International Association for Machine Translation.

Niraj Aswani and Robert Gaizauskas. A hybrid approach to align sentences and words in English-Hindi parallel corpora. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 57–64, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Michael Auli, Michel Galley, and Jianfeng Gao. Large-scale expected bleu training of phrase-based reordering models. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1250–1260. Association for Computational Linguistics, 2014.

Necip Fazil Ayan, Bonnie J. Dorr, and Christof Monz. NeurAlign: Combining word alignments using neural networks. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 65–72, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

Srinivas Bangalore, Patrick Haffner, and Stephan Kanthak. Statistical machine translation through global lexical selection and sentence reconstruction. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 152–159, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

Phil Blunsom and Trevor Cohn. Discriminative word alignment with conditional random fields. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 65–72, Sydney, Australia, July 2006. Association for Computational Linguistics.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, 1992.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz Josef Och, and Jeffrey Dean. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, 2007.

P. Brown, V. Della Pietra, S. Della Pietra, and R. Mercer. The mathematics of statistical machine. translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Peter F. Brown, John Cocke, Stephen A. Della-Pietra, Vincent J. Della-Pietra, Frederick Jelinek, Robert L. Mercer, and Paul Rossin. A statistical approach to language translation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 1988.

Peter F. Brown, Jennifer C. Lai, and Robert L. Mercer. Aligning sentences in parallel corpora. In *Proceedings of the 29th Annual Meeting of the Association of Computational Linguistics (ACL)*, 1991.

Tim Buckwalter. Buckwalter Arabic Morphological Analyzer Version 2.0. Linguistic Data Consortium (Catalog No. LDC2004L02), 2004.

Chris Callison-Burch, David Talbot, and Miles Osborne. Statistical machine translation with word- and sentence-aligned parallel corpora. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 175–182, Barcelona, Spain, July 2004.

Marine Carpuat and Dekai Wu. Context-dependent phrasal translation lexicons for statistical machine translation. In *Proceedings of the MT Summit XI*, Copenhagen, Denmark, September 2007. European Association for Machine Translation.

Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. Training and testing low-degree polynomial data mappings via linear svm. *Journal of Machine Learning Research*, 2:1471–1490, April 2010.

S. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. Technical report, Computer Science Group, Harvard University, 1998.

Colin Cherry. Improved reordering for phrase-based translation using sparse features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 22–31, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Colin Cherry and Dekang Lin. Soft syntactic constraints for word alignment through discriminative training. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 105–112, Sydney, Australia, July 2006. Association for Computational Linguistics.

David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2), 2007.

Wei Chu and Zoubin Ghahramani. Gaussian processes for ordinal regression. *The Journal of Machine Learning Research*, 6:1019–1041, December 2005. ISSN 1532-4435.

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 19th International Conference on Machine Learning (ICML 2002)*, pages 744–751, 2002.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9:1775–1822, June 2008. ISSN 1532-4435.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, September 1995.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-78019-5.

J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

Adrià de Gispert, Gonzalo Iglesias, Graeme Blackwood, Eduardo R. Banga, and William Byrne. Hierarchical phrase-based translation with weighted finite-state transducers and shallow-n grammars. *Computational Linguistics*, 36(3), 2010.

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 138–145, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

Andreas Eisele and Yu Chen. Multiun: A multilingual corpus from united nation documents. In Daniel Tapias, Mike Rosner, Stelios Piperidis, Jan Odjik, Joseph Mariani, Bente Maegaard, Khalid Choukri, and Nicoletta Calzolari (Conference Chair), editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation*, pages 2868–2872. European Language Resources Association (ELRA), 5 2010.

Jessica Enright and Grzegorz Kondrak. A fast method for parallel document identification. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 29–32, Rochester, New York, April 2007. Association for Computational Linguistics.

Pablo A. Estévez, Michel Tesmer, Claudio A. Perez, and Jacek M. Zurada. Normalized mutual information feature selection. *Trans. Neur. Netw.*, 20(2):189–201, February 2009.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Bassam Farran and Craig Saunders. Voted spheres: An online, fast approach to large scale learning. In *Proceedings of the International Conference on Advanced Information Networking and Applications Workshops*, pages 744–749. IEEE, 2009.

Marcello Federico and Mauro Cettolo. Efficient handling of n-gram language models for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 88–95, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Minwei Feng, Jan-Thorsten Peter, and Hermann Ney. Sequence labeling-based reordering model for phrase-based smt. In *International Workshop on Spoken Language Translation*, pages 260–267, Hong Kong, December 2012.

Alex Franz and Thorsten Brants. All our n-gram are belong to you. `http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html`, 2006. Last accessed 4 April 2014.

Thilo-Thomas Friess, Nello Cristianini, and Colin Campbell. The kernel-adatron algorithm: a fast and simple learning procedure for support vector machines. In *Machine Learning: Proceedings of the Fifteenth International Conference*. Morgan Kaufmann, 1998.

Ken'ichi Fukushima, Kenjiro Taura, and Takashi Chikayama. A fast and accurate method for detecting English-Japanese parallel texts. In *Proceedings of the Workshop on Multilingual Language Resources and Interoperability*, pages 60–67, Sydney, Australia, July 2006. Association for Computational Linguistics.

William A. Gale and Kenneth Ward Church. A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19:75–102, 1993.

Michel Galley and Christopher D. Manning. A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Honolulu, Hawaii, October 2008. Association for Computational Linguistics.

Ulrich Germann. Building a statistical machine translation system from scratch: How much bang for the buck can we expect? In *Workshop on Data-Driven Machine Translation at 39th Annual Meeting of the Association of Computational Linguistics (ACL)*, 2001.

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

Jesús Giménez and Lluís Màrquez. Context-aware discriminative phrase selection for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 159–166, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003.

Zhongjun He, Yao Meng, and Hao Yu. Maximum entropy based phrase reordering for hierarchical phrase-based translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 555–563. Association for Computational Linguistics, 2010.

Mark Hopkins and Jonathan May. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 408–415, 2008.

Matthias Huck, Stephan Peitz, Markus Freitag, and Hermann Ney. Discriminative reordering extensions for hierarchical phrase-based machine translation. In *Proceedings of 16th Annual Conference of the European Association for Machine Translation*, pages 313–320, Trento, Italy, May 2012.

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. Hierarchical phrase-based translation with weighted finite state transducers. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 433–441, Boulder, Colorado, June 2009. Association for Computational Linguistics.

Abraham Ittycheriah and Salim Roukos. A maximum entropy word aligner for Arabic-English machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 89–96, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, 2006.

S. Sathiya Keerthi, Sellamanickam Sundararajan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A sequential dual method for large scale multi-class linear SVMs. In *Proceedings of the Forteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 408–416, 2008.

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Accoustics, Speech and Signal Processing*, volume 1, 1995.

Kevin Knight. Automating knowledge acquisition for machine translation. *AI Magazine*, 18(4), 1997.

Kevin Knight. A statistical MT tutorial workbook. *unpublished*, 1999.

P. Koehn. *Statistical Machine Translation*. Cambridge University Press, 2010a.

Philipp Koehn. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of 6th Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 115—124, Washington DC, 2004a.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July 2004b. Association for Computational Linguistics.

Philipp Koehn. An experimental management system. *The Prague Bulletin of Mathematical Linguistics*, 94:87–96, 2010b.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proc. of the International Workshop on Spoken Language Translation*, October 2005.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Christopher J. Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 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, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Philipp Koehn and Kevin Knight. Empirical methods for compound splitting. In *Proceedings of Meeting of the European Chapter of the Association of Computational Linguistics (EACL)*, 2003.

Philipp Koehn and Christof Monz. Shared task: Statistical machine translation between European languages. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 119–124, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Shankar Kumar and William Byrne. Local phrase reordering models for statistical machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 161–168,

Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

Guy Lebanon and John D. Lafferty. Boosting and maximum likelihood for exponential models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 447–454. MIT Press, 2002.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5: 361–397, December 2004.

Bo Li and Juan Liu. Mining Chinese-English parallel corpora from the web. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP)*, 2008.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia, July 2006. Association for Computational Linguistics.

Huan Liu and Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. In Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

Adam Lopez. Statistical machine translation. *ACM Computing Surveys*, 40(3):8:1–8:49, August 2008. ISSN 0360-0300.

Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Identifying suspicious urls: An application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 681–688, 2009.

David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.

Hemali Majithia, Philip Rennart, and Evelyne Tzoukermann. Rapid ramp-up for statistical machine translation: Minimal training for maximal coverage. In *Proceedings of the Tenth Machine Translation Summit (MT Summit X)*, Phuket, Thailand, September 2005.

Joel Martin, Howard Johnson, Benoit Farley, and Anna Maclachlan. Aligning and using an English-Inuktitut parallel corpus. In Rada Mihalcea and Ted Pedersen, editors, *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, Edmonton, Alberta, Canada, May 31 2003. Association for Computational Linguistics.

Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of the AAAI'98 Workshop on Learning for Text categorization*, 1998.

Peter McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society*, 42:109–142, 1980.

Arul Menezes and Chris Quirk. Microsoft research treelet translation system: IWSLT evaluation. In *Proc. of the International Workshop on Spoken Language Translation*, October 2005.

Robert C. Moore. A discriminative framework for bilingual word alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 81–88, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

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

Dragos Stefan Munteanu, Alexander Fraser, and Daniel Marcu. Improved machine translation performance via parallel sentence extraction from comparable corpora. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, 2004.

Dragos Stefan Munteanu and Daniel Marcu. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4), 2005.

Dragos Stefan Munteanu and Daniel Marcu. Extracting parallel sub-sentential fragments from non-parallel corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 81–88, Sydney, Australia, July 2006. Association for Computational Linguistics.

Vinh Van Nguyen, Akira Shimazu, Minh Le Nguyen, and Thai Phuong Nguyen. Improving a lexicalized hierarchical reordering model using maximum entropy. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*, Ottawa, Ontario, Canada, 2009. International Association for Machine Translation.

Yizhao Ni. *Beyond multi-class – structured learning for machine translation.* PhD thesis, The University of Southampton, 2010.

Yizhao Ni, Mahesan Niranjan an Craig Saunders, and Sandor Szedmak. Distance phrase reordering for moses - user manual and code guide. Technical report, School of Electronics and Computer Science, University of Southampton, Southampton, UK, 2010a.

Yizhao Ni, Craig Saunders, Sandor Szedmak, and Mahesan Niranjan. The application of structured learning in natural language processing. *Machine Translation Journal*, 24(2):71–85, 2010b.

Yizhao Ni, Craig Saunders, Sandor Szedmak, and Mahesan Niranjan. Exploitation of machine learning techniques in modelling phrase movements for machine translation. *Journal of Machine Learning Research*, 12:1–30, February 2011. ISSN 1532-4435.

Jan Niehues and Stephan Vogel. Discriminative word alignment via alignment matrix modeling. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 18–25, Columbus, Ohio, June 2008. Association for Computational Linguistics.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, 2003.

Franz Josef Och and Hermann Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association of Computational Linguistics (ACL)*, 2000.

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics (ACL)*, 2002.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52, 2003.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods*, pages 185–208. MIT Press, 1999.

Chris Quirk, Raghavendra Udupa, and Arul Menezes. Generative models of noisy translations with applications to parallel fragment extraction. In *Proceedings of the MT Summit XI*, Copenhagen, Denmark, September 2007. International Association for Machine Translation.

Philip Resnik. Mining the web for bilingual text. In *Proceedings of the 37th Annual Meeting of the Association of Computational Linguistics (ACL)*, 1999.

Simon Rogers and Mark Girolami. *A First Course in Machine Learning*. Chapman & Hall/CRC, 1st edition, 2011.

Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. Kernel-based earning of hierarchical multilabel classification models. *The Journal of Machine Learning Research*, pages 1601–1626, 2006.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 807–814, 2007.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

Libin Shen, Anoop Sarkar, and Franz Josef Och. Discriminative reranking for machine translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, 2004.

Jonathan Slocum. A survey of machine translation: its history, current status, and future prospects. *Computational Linguistics*, 11(1):1–17, 1985.

Alexander Statnikov, Constantin Aliferis, Ioannis Tsamardinos, Douglas Hardin, and Shawn Levy. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, 21(5):631–643, 2005.

Ben Taskar, Lacoste-Julien Simon, and Dan Klein. A discriminative matching approach to word alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 73–80, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

Christoph Tillmann. A unigram orientation model for statistical machine translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, 2004.

Christoph Tillmann and Tong Zhang. A localized prediction model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for*

Computational Linguistics (ACL'05), pages 557–564, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Marco Turchi, Tijl De Bie, Cyril Goutte, and Nello Cristianini. Learning to translate: A statistical and computational analysis. *Advances in Artificial Intelligence*, 2012, January 2012.

Sriram Venkatapathy and Aravind Joshi. Discriminative word alignment by learning the alignment structure and syntactic divergence between a language pair. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 49–56, Rochester, New York, April 2007. Association for Computational Linguistics.

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. Word-sense disambiguation for machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 771–778, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

Zhuoran Wang and John Shawe-Taylor. Kernel regression framework for machine translation: UCL system description for WMT 2008 shared translation task. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 155–158, Columbus, Ohio, June 2008. Association for Computational Linguistics.

Zhuoran Wang, John Shawe-Taylor, and Sandor Szedmak. Kernel regression based machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 185–188, Rochester, New York, April 2007. Association for Computational Linguistics.

Steve Webb, James Caverlee, and Calton Pu. Introducing the webb spam corpus: Using email spam to identify web spam automatically. In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS)*, Mountain View, California, USA, July 2006.

Hua Wu and Haifeng Wang. Boosting statistical word alignment. In *Proceedings of the Tenth Machine Translation Summit (MT Summit X)*, Phuket, Thailand, September 2005. International Association for Machine Translation.

Hua Wu, Haifeng Wang, and Zhanyi Liu. Boosting statistical word alignment using labeled and unlabeled data. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 913–920, Sydney, Australia, July 2006. Association for Computational Linguistics.

Bing Xiang, Niyu Ge, and Abraham Ittycheriah. Improving reordering for statistical machine translation with smoothed priors and syntactic features. In *Proceedings of*

SSST-5, Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation, pages 61–69, Portland, Oregon, USA, 2011. Association for Computational Linguistics.

Deyi Xiong, Qun Liu, and Shouxun Lin. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 521–528, Sydney, July 2006. Association for Computational Linguistics.

Hsiang-Fu Yu, Cho-Jui Hsieh, Kai-Wei Chang, and Chih-Jen Lin. Large linear classification when data cannot fit in memory. *ACM Transactions on Knowledge Discovery from Data*, 5(4):23:1–23:23, February 2012.

Hsiang-Fu Yu, Fang-Lan Huang, and Chih-Jen Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1-2):41–75, October 2011. ISSN 0885-6125.

Richard Zens and Hermann Ney. Discriminative reordering models for statistical machine translation. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 55–63, New York City, June 2006. Association for Computational Linguistics.

Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2): 179–214, 2004.