Masterarbeit im Fach Informatik
Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik 6
Prof. Dr.-Ing. Hermann Ney

Phrase Table Smoothing with Word Classes

30. Juli 2015

vorgelegt von:
Yunsu Kim
Matrikelnummer 315779

Gutachter:
Prof. Dr.-Ing. Hermann Ney
Prof. Dr. Bastian Leibe

Betreuer:
Dipl.-Inform. Andreas Guta
Dipl.-Inform. Jörn Wübker

Erklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Textauszüge und Grafiken, die sinngemäß oder wörtlich aus veröffentlichten Schriften entnommen wurden, sind durch Referenzen gekennzeichnet.

Aachen, 30. Juli 2015

Yunsu Kim

Abstract

Phrase-based statistical machine translation (SMT) has a severe data sparsity problem in computing translation probabilities. In this thesis, we present and analyze various usages of word classes to alleviate the sparsity and improve the translation quality of a phrase-based SMT system.

First of all, we propose a novel smoothing formulation of phrase translation models, using word classes on the word level within a phrase. Secondly, we modify the standard phrase-based decoder to utilize word class phrases as additional translation options. Finally, we investigate the word alignments trained from word classes.

The performance of our proposed approaches is measured on three different translation tasks to prove their broad applicability. The experiments show that our smoothed translation model is comparable to the state-of-the-art word class models with a smaller number of features. In addition, our modified decoder significantly reduces the out-of-vocabulary rate and enhances the overall translation quality in both automatic metrics and manual evaluation.

We also make an extensive comparison among different word class mappings in terms of their performance in phrase-based SMT. Our results reveal that only the number of classes affects the translation quality of our proposed methods, regardless of the type of clustering algorithms and other parameters for estimating word classes.

Contents

Chapter 1

Introduction

Statistical machine translation (SMT) has achieved an outstanding success over the last decade by adopting phrases as atomic translation units. Phrase-to-phrase translation simplifies the modeling of local reorderings and multi-word lexical choices, producing more natural translations than word-based translation systems [Zens & Och$^+$ 02, Och 02, Koehn & Och$^+$ 03]. However, since phrase vocabulary is much larger than word vocabulary, much larger training data is needed to obtain reliable statistics for phrase-based SMT. Unfortunately, for many language pairs, it is highly demanding to collect a sufficient amount of bilingual corpora for training robust phrase translation models.

A fundamental solution to the sparsity problem in many natural language processing (NLP) tasks is to reduce the vocabulary size. Most probability models for NLP are built on a huge, discrete word vocabularies. By mapping words onto a smaller label space, the models can be effectively trained to have denser distributions. The label also enriches the resulting systems by conveying additional information for each word. The examples of such labels include part-of-speech (POS) tags, morphological stems or automatically generated word classes. They have been used in language modeling [Brown & deSouza$^+$ 92], named entity recognition (NER) [Miller & Guinness$^+$ 04, Liang 05] or parsing [Koo & Carreras$^+$ 08] for smoothing sparse word models. In this thesis, we investigate the application of word classes to phrase-based SMT.

A word class is a grouping of words with syntactic/semantic similarity and can be automatically determined by clustering algorithms. Unlike other word labels, its estimation is fully unsupervised; the clustering does not require model training with language-specific annotations. This is particularly appealing to SMT tasks where various languages are handled at the same time. Moreover, the structure and size of a word class vocabulary can be arbitrarily adjusted by the corresponding clustering parameters, which makes it possible to control the degree of smoothing. Traditionally in word-based SMT, word classes are used to smooth the alignment models of IBM models 4 and 5 [Brown & Pietra$^+$ 93].

For phrase-based SMT, this work proposes a novel formulation of smoothing phrase translation models based on word classes. We build a smaller phrase vocab-

ulary by replacing each word in a phrase with its corresponding word class. Our model is unique in that it replaces one word at a time to construct phrases with both words and word classes. We evaluate our method on a small dataset (IWSLT 2012 German→English) and two large datasets (WMT 2014 English→German, WMT 2015 English→Czech), proving its comparable performance to the state-of-the-art with a smaller number of features.

We also empirically study word class smoothing of language models in a phrase-based SMT system. Our experiments show that word class language models consistently improve the baseline system in combination with the smoothed translation models. We verify that their performance can be further enhanced by refactorizing with the class membership probability, i.e. the conditional probability of a word for a given class.

Apart from the modeling, we develop a class-based paraphrasing method which can be easily integrated into phrase-based decoding. The idea is to substitute a word in a phrase with another word if both words are in the same word class. In this way, we obtain new phrase pairs which are not extracted from the training corpus. We propose a simple modification of the standard phrase-based decoder to use these paraphrases as extra translation options. This method considerably improves the translation quality, according to our results.

We additionally present a method for training word alignments from word classes. Before running a word alignment tool, we preprocess a bilingual training corpus by replacing every word with its respective word class. The acquired alignment is merged with the original word alignment or directly applied to compute word class model scores.

Our experiments are carried out with various word class mappings, which differ in clustering algorithm, clustering iterations, initial clusterings and the number of classes. The motivation is to find the optimal word class mapping which maximizes the performance of our proposed methods. The results reveal that the word class estimation only the number of classes has an effect on the translation quality of some word class models.

Related Work

There have been numerous attempts on utilizing additional word labels in phrase-based SMT. An early group of work focuses on pre- and postprocessing bilingual corpora with syntactic labels, compensating for the difference in word order between source and target languages [Xia & McCord 04, Collins & Koehn$^+$ 05, Popovič & Ney 06].

Another approach is to insert separate steps in the translation process, which use word labels to improve the model accuracy and refine the search space. The work in this direction is biased towards inflection modeling with morphological

labels [Toutanova & Suzuki$^+$ 08, Fraser & Weller$^+$ 12, Chahuneau & Schlinger$^+$ 13], for which one needs to train linguistic analyzers per language. The integration of these methods into current phrase-based systems requires substantial modifications of the training and translation pipelines.

[Yang & Kirchhoff 06] extract hierarchical paraphrases using a morphological analysis and use them as translation options. Their approach normalizes the translation scores of the paraphrases along with original phrases, eventually having a discounting effect. Our decoder modification is inspired by their work but differs in the following aspects:

- We use automatically clustered word classes to avoid any language-specific analysis.
- Besides the application in decoding, we build an additional translation model with the same paraphrasing concept.
- We define a separate parameter for weighting the score of the paraphrases, which is directly tuned with respect to translation metrics.

[Koehn & Hoang 07] introduce a generic framework for integrating multiple word-level labels as factors into the standard phrase-based SMT process. It consists of the following three steps:

1. The mapping from words to labels
2. The Label-to-label translation
3. The generation of words from labels

Assuming probabilistic mappings between words and labels, the first and third steps imply a combinatorial explosion in the number of phrase translation rules. [Koehn & Hoang 07] solve this problem by aggressive pruning.

Recent research is devoted to exploit word labels with only a little change of the existing system, e.g. by adding new features to the existing modeling, or no change at all. [Green & DeNero 12] design a syntax agreement model on the target side using morpho-syntactic labels. [Cherry 13] use word classes to reformulate sparse reordering features. [Wuebker & Peitz$^+$ 13] train the standard translation, language, and reordering models on word classes. They show a simplified case of [Koehn & Hoang 07] by adopting hard assignment from words to labels, dramatically reducing the phrase table size and implementation effort. Our work is also based on their simplified assumption, but presents a more efficient and elaborate translation model. A comparative study on language models with additional labels is found in [Bisazza & Monz 14].

In a completely different direction, [Foster & Kuhn$^+$ 06] apply language model smoothing techniques to phrase translation models. They treat a phrase as a single

entity of a phrase n-gram and perform backoff by removing the entity from the conditioning part, e.g. $p(\tilde{f}|\tilde{e}) \rightarrow p(\tilde{f})$. Dealing with phrase pairs, their formulation is restricted only to bigrams. Also, it does not involve further analysis of the inside of phrases. According to our experience, their performance is very limited, especially when a word-based lexicon model is already being used.

The remainder of this thesis is structured as follows. In the next chapter, we first review the entire phrase-based SMT pipeline and identify room for improvement with word classes. Chapter 3 explains the concept of word classes and possible algorithms for estimating them. Our proposed methods are thoroughly described in Chapter 4. The experimental settings and results are shown in Chapter 5. We conclude with a summary and future work in Chapter 6.

Chapter 2

Phrase-Based Statistical Machine Translation

This chapter describes each component of the phrase-based SMT process. We explain its core concepts with mathematical formulations and analyze the points to be enhanced using word classes.

2.1 Word Alignments

For a translation task, a pair of sentences in two different languages is considered. One is a sentence to be translated (source sentence) and the other is its translation (target sentence). Such pair is called a bilingual sentence, represented as sequences of words:

$$f_1^J = f_1, ..., f_j, ..., f_J \tag{2.1}$$

$$e_1^I = e_1, ..., e_i, ..., e_I \tag{2.2}$$

where f_j denotes a source word and e_i a target word. J and I are the lengths of the source sentence and the target sentence, respectively. In SMT, a set of bilingual sentences is needed to train the models.

Before the training, one must know which target word is the translation of which source word in each bilingual sentence. In other words, each target word should be aligned to the related source words, and vice versa. Figure 2.1 shows an example of word alignments. Our notation for source-to-target alignments is

$$a_1^J = a_1, ..., a_j, ..., a_J \ , \tag{2.3}$$

where a_j denotes the target word positions aligned to f_j:

$$a_j \subseteq \{1, ..., I\} \quad \text{or} \quad a_j = \{0\} \tag{2.4}$$

Position 0 stands for an empty target word, thus $\{0\}$ indicates that a source word is not aligned to any target word.

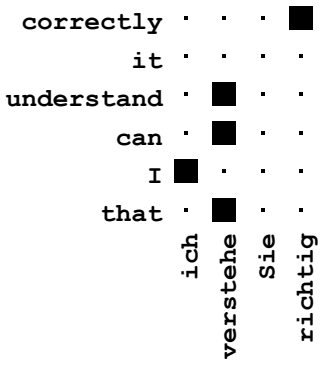


Figure 2.1: Word alignments of a German-English sentence pair, learned with GIZA++. The German word "Sie" and the English word "it" are unaligned. The English word "that" is wrongly aligned to the German word "verstehe" which has the meaning of "understand".

In most cases, word alignments are not accompanied with a bilingual corpus. Annotating bilingual sentence pairs with word alignments requires an enormous human effort. Alternatively, word alignments can be automatically learned using statistical models. GIZA++ is the most widely used software for this purpose, based on the IBM models and the hidden Markov model [Och & Ney 03]. We use GIZA++ throughout all experiments of this thesis.

The quality of statistically trained word alignments is far behind the human work. It is considerably varying with model selection and parameters, and is subjective depending on the context. To remedy this, several different word alignments are often merged using heuristics, e.g. union or intersection.

An efficient method for producing a meaningfully different word alignment is to train alignment models with a modified text. For example, the same corpus with a reversed word order for each sentence feeds the models with word collocations in the inverse direction. Another idea is to replace each word with other entities, which deliver additional information on the word.

2.2 Phrase Extraction

For phrase translation, one should also learn phrase alignments from bilingual sentences, i.e. how multiple target words are aligned to multiple source words. The aligned *phrase pairs* are used as basic translation rules of a phrase-based SMT system.

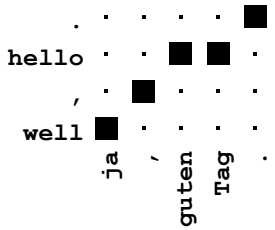
For mathematical/implementational simplicity, the early phrase-based SMT systems [Zens & Och$^+$ 02,Och 02,Koehn & Och$^+$ 03] define a phrase to be a continuous sequence of words. This restriction is relaxed in later research by allowing hierarch-

ical [Chiang 07] and discontinuous phrases [Galley & Manning 10]. Within this work, we follow the original definition of phrases. Our general notations for source and target phrases are:

$$\tilde{f} = f_{j_1}, ..., f_{j_2} \tag{2.5}$$

$$\tilde{e} = e_{i_1}, ..., e_{i_2} \tag{2.6}$$

where j_1 and i_1 are the begin positions and j_2 and i_2 are the end positions of the phrases. Based on this definition, we use extraction heuristics of [Och 02] and extract only those phrase pairs whose words are aligned only within the phrase boundaries. An example of the phrase extraction is shown in Figure 2.2 and Table 2.1.



Source phrase	Target phrase
ja	well
ja ,	well ,
ja , guten Tag	well , hello
ja , guten Tag .	well , hello .
,	,
, guten Tag	, hello
, guten Tag .	, hello .
guten Tag	hello
guten Tag .	hello.
.	.

Figure 2.2 & Table 2.1: [Zens 08] A bilingual sentence and the extracted phrase pairs.

All the extracted phrase pairs are stored in a phrase table along with their model scores. The phrase table is queried during the actual translation step (Section 2.5). A target translation e_1^I is generated by applying the phrase pairs to a given source sentence f_1^J. In this respect, a sentence pair (f_1^J, e_1^I) can be seen as a sequence of phrase pairs $(\tilde{f}_1^K, \tilde{e}_1^K)$. We introduce the notation of phrase segmentation for each $k = 1, ..., K$

$$k \rightarrow s_k := (i_k; b_k, j_k), \tag{2.7}$$

where i_k is the end position of the k-th target phrase, and b_k and j_k are the begin and end positions of the k-th source phrase, respectively. Using these indices, we denote each phrase pair $(\tilde{f}_k, \tilde{e}_k)$ as:

$$\tilde{f}_k := f_{b_k},\ ...,\ f_{j_k} \tag{2.8}$$

$$\tilde{e}_k := e_{i_{k-1}+1},\ ...,\ e_{i_k} \tag{2.9}$$

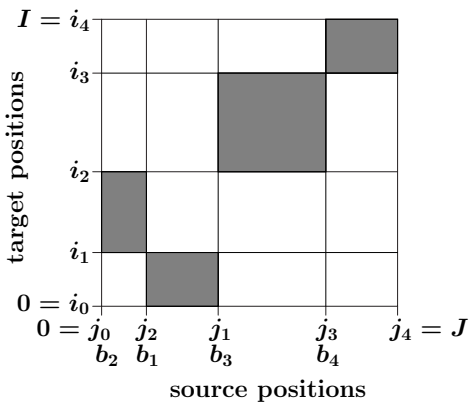


Figure 2.3: [Zens 08] Illustration of phrase segmentation.

This notation implicitly includes phrase reordering information. Figure 2.3 shows an example segmentation of a bilingual sentence. Note that there are no overlapping phrase pairs, since no source word is translated twice.

2.3 Models

Given a source sentence, an SMT system scores its target translations with probability models which represent aspects of translation quality. Translation is a highly complex task; to accurately translate a sentence, diverse factors must be considered at the same time, e.g. word/phrase lexicon, semantics, grammar, etc. *Log-linear* modeling is a suitable probabilistic framework for this purpose [Berger & Pietra$^+$ 96, Och & Ney 02]. It can effectively combine hundreds or thousands of features. Using log-linear framework, the probability of a target translation given a source sentence can be modeled as:

$$p(e_1^I|f_1^J) = \max_{K,s_1^K} \frac{\exp\left(\sum_{m=1}^M \lambda_m h_m(e_1^I, s_1^K; f_1^J)\right)}{\sum_{e_1'^{I'}, s_1'^{K'}} \exp\left(\sum_{m=1}^M \lambda_m h_m(e_1'^{I'}, s_1'^{K'}; f_1^J)\right)} \tag{2.10}$$

where h_m is an individual model and λ_m is the corresponding model weight. Each model is built upon the statistics gathered from training corpora. Note that h_m is not necessarily a probability distribution; it can be any type of a scoring function, e.g. an unnormalized count or an extremely sparse binary feature. The scores are scaled by λ_m and easily integrated into the overall probability model. Due to its flexibility, log-linear models are adopted in solving various NLP problems, including part-of-speech tagging [Ratnaparkhi 96], named entity recognition [Borthwick & Sterling$^+$ 98], and language modeling [Khudanpur & Wu 00]. In the follow-

ing sections, we present the standard set of features used in our log-linear model combination.

Phrase translation model

The most essential feature in phrase-based SMT is the phrase translation model. Given a bilingual sentence and its phrase segmentation, the phrase translation model is defined as follows:

$$h_{Phr}(e_1^I, s_1^K; f_1^J) = \sum_{k=1}^{K} \log p_{Phr}(\tilde{f}_k|\tilde{e}_k) \tag{2.11}$$

where the phrase translation probability for each phrase pair $(\tilde{f}, \tilde{e})$ is estimated using relative frequencies:

$$p_{Phr}(\tilde{f}|\tilde{e}) = \frac{N(\tilde{f}, \tilde{e})}{N(\tilde{e})} \tag{2.12}$$

For a symmetric modeling, the model is also built in the inverse translation direction using $p_{Phr}(\tilde{e}|\tilde{f})$.

Relative frequencies are prone to overfitting. It readily produces unreliable scores for infrequent events, e.g. long phrases or phrases containing rare words. For morphologically rich languages, e.g. Arabic or German, even common phrases tend to have low counts; since they appear in many different forms depending on the syntactic context, distributing the counts over several morphological variations.

Word-based lexicon model

To compensate for the sparsity of the phrase translation model, each phrase is decomposed into words and their word translation probabilities are calculated. A word lexicon has much smaller vocabulary than phrase lexicon, so its counts are more robust given the same amount of training data. We define a word-based feature function by summing up all word translation probabilities within the phrase boundary [Zens 08]:

$$h_{Lex}(e_1^I, s_1^K; f_1^J) = \sum_{k=1}^{K} \log \prod_{j=b_k}^{j_k} \frac{1}{|\tilde{e}_i|} \sum_{i=i_{k-1}+1}^{i_k} p(f_j|e_i) \tag{2.13}$$

which is analogous to IBM model 1 [Brown & Pietra$^+$ 93]. We use relative frequencies to compute $p(f|e)$ just as the phrase translation model. The probability sum over the target positions is normalized by the length of the target phrase $\tilde{e}_i$, otherwise longer target phrases unintentionally have higher scores. The inverse model for the source-to-target direction is defined in a similar way with $p(e|f)$.

The word-based lexicon model was proposed with the birth of phrase-based SMT [Koehn & Och$^+$ 03], and still widely accepted as a standard feature in recent phrase-based SMT systems. The model has, however, two main drawbacks. First, local contexts are too coarsely considered. Neighboring words are regarded only as triggers to a single word on the opposite side, and the relation between them, e.g. lexical preferences in the phrase context and local reorderings, are ignored. Second, the model still overfits for rare words, although it effectively handles rare phrases.

Language Models

The translation of a source sentence can be seen as the generation of a target sentence given a set of clues. In all language generation problems, e.g. automatic speech recognition, language model is a core component to ensure the generated sentence to be actually probable in terms of syntax and semantics. SMT systems also utilize a language model, which is estimated as an n-gram Markov model of the target words:

$$h_{LM}(e_1^I, s_1^K; f_1^J) = \sum_{i=1}^{I} \log p(e_i | e_{i-n+1}^{i-1}) \tag{2.14}$$

n-gram models also suffer from the sparsity problem. It is addressed by backing off to lower-order n-gram probabilities. For a more precise modeling, multiple language models can be combined together, each of which is trained with different lexical entities such as POS tags or word classes [Koehn & Hoang 07].

Reordering Models

Languages have different word orders. For instance, in Korean sentences, an object occurs before the corresponding verb, while in English the order is swapped. To generate grammatically correct translations, SMT systems should rearrange the translated phrases in the target language. Such reorderings can be simply modeled with the following distortion penalty:

$$h_{DP}(e_1^I, s_1^K; f_1^J) = \sum_{k=1}^{K} q_{Dist}(b_k, j_{k-1}) \tag{2.15}$$

with

$$q_{Dist}(j, j') := \begin{cases} |j - j' + 1| & \text{if } |j - j' + 1| < D \\ \infty & \text{else} \end{cases} \tag{2.16}$$

which is the distance from the begin position of the current phrase to the end position of the last phrase. The parameter D controls the maximum distortion length allowed. More complicated models are also used to capture exact reordering behaviors with respect to specific words [Tillmann 04, Galley & Manning 08]. Those lexicalized models have severe sparsity problem, since the model complexity is multiplied by the size of the word vocabulary.

2.4 Training

After the estimation of individual features h_m, the scaling factors λ_m have to be optimized such that the resulting log-linear model (2.10) well explains the translation between two languages. For this purpose, a separate bilingual corpus is used, which is disjunct from the one used for training the features. It is called a development set, denoted by $(\mathcal{F}, \mathcal{R})$, where $\mathcal{F}$ is a sequence of source sentences and $\mathcal{R}$ is the corresponding reference translations. Using a development set, we expect to prevent the log-linear model from overfitting to training data.

The minimum error rate training (MERT) [Och 03] is a state-of-the-art algorithm for training λ_m with respect to a final evaluation metric, e.g. BLEU [Papineni & Roukos$^+$ 02]. It optimizes the following criterion:

$$\hat{\lambda}_1^M = \arg\min_{\lambda_1^M} \left\{ E(\mathcal{R}, \hat{\mathcal{E}}(\mathcal{F}, \lambda_1^M)) \right\} \tag{2.17}$$

where $\hat{\mathcal{E}}$ is the best hypothesis translation of $\mathcal{F}$ and $E(\mathcal{R}, \hat{\mathcal{E}})$ is the error of the hypothesis for a given metric. An SMT system produces N-best translations with different set of parameters λ_1^M, and select the set whose hypothesis has the minimal error. The error measure E can be constructed with any evaluation metric for SMT. We train λ_1^M using MERT with the BLEU metric in all our experiments.

The main disadvantage of MERT is that it does not reliably optimize the scaling factors when the number of features M is too large, i.e. over approximately 30. It is because the algorithm is not regularized and use non-convex loss function. A current research direction is to develop a scalable training method which accommodates hundreds or thousands of features, which is especially useful for a rich set of binary features [Chiang & Knight$^+$ 09, Hopkins & May 11, Green & Wang$^+$ 13]. Since our system uses MERT, it is crucial to keep the number of features reasonably small.

2.5 Decoding

A phrase-based SMT system generates the best translation of a given source sentence according to the utilized model. It is referred to as decoding, since a source sentence can be regarded as the unknown, encrypted text and the corresponding

target sentence as the readable, decrypted text. Decoding is achieved by the Bayes decision rule:

$$f_1^J \longmapsto \hat{e}_1^{\hat{I}}(f_1^J) = \underset{I,e_1^I}{\operatorname{argmax}} \max_{K,s_1^K} \left\{ \sum_{m=1}^{M} \lambda_m h_m(e_1^I, s_1^K; f_1^J) \right\} \tag{2.18}$$

The Bayes decision rule performs a maximization over all possible target sentences e_1^I and over all possible phrase segmentations s_1^K, which is infeasible in practice. Therefore, the search space is restricted to contain only highly probable translations, discarding numerous unlikely candidates. Here, we present a standard procedure of phrase-based decoders.

2.5.1 Phrase Matching

Prior to the actual search, it is beneficial to mark all the phrase pairs that can actually be used for the given source sentence. We iterate over all possible source phrases in the sentence and find the matching phrase table entries, which are loaded into memory for convenient use. The more matching is done, the more phrase translation options are available.

Note that the matched phrase pairs are only the ones extracted from a bilingual training corpus. If the training data does not contain enough examples, one might not have necessary translation options for the given source sentence. The straightforward solution is to increase the training corpus size, which may not be available for low-resource language pairs. An alternative is to manipulate the extracted phrase pairs to obtain extra phrase pairs. Section 4.2 explains such a method using word classes.

2.5.2 Search Graph

In search, we exploit dynamic programming [Bellman 57] to find the best hypothesis translation. We begin with an empty hypothesis and expand it incrementally by appending one target phrase at a time. For each partial hypothesis, we first check which source words are not translated yet, and choose a continuous sequence $f_{b_k}^{j_k} = \tilde{f}_k$ to be translated next. We retrieve the list of possible target translations $\tilde{e}_k$ for $\tilde{f}_k$ from the subset of matching phrases, which are then used to expand the hypothesis. To keep track of the words that have been translated, we maintain a coverage set $O \subseteq \{1, ..., J\}$ of source positions, which is updated at the end of every expansion. This set ensures that each source word is translated exactly once. Once all the source positions have been covered, a complete hypothesis translation is made.

The search procedure can be represented as a directed graph. The nodes are labeled with coverage sets C, and the edges are labeled with tuples $(\tilde{e}_k, b_k, j_k)$ which

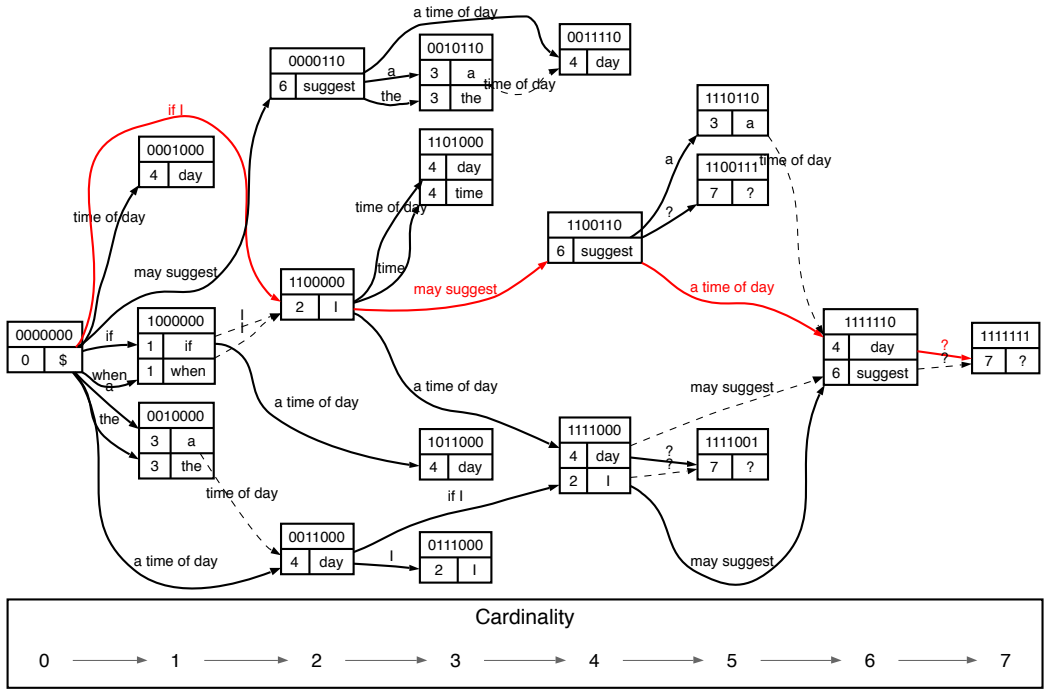


Figure 2.4: [Zens 08] The search graph for translating the German sentence "`Wenn ich eine Uhrzeit vorschlagen darf ?`". Each node is labeled with coverage set C (as a bit vector), the end position j of the previous source phrase, and the language model history (here: bigram). Dashed edges indicate recombination: a path is ignored when another path has a higher score with the same translation. The optimal path ("`If I may suggest a time of day ?`") is marked in red. Scores are omitted within this figure.

indicate each expansion. Accordingly, a hypothesis translation is a path through the graph, starting from the root node $C = \emptyset$.

In the log-linear framework, a hypothesis score can be easily split into phrase-level scores, i.e. sum over individual expansions. At each expansion $(\tilde{e}_k, b_k, j_k)$, we update the score of a hypothesis on the fly by simply adding up the score of that expansion. For most features, the expansion score depends only on the current phrase pair $(\tilde{f}_k, \tilde{e}_k)$. Thus, in the search graph, each edge contains the features scores of the corresponding expansion. On the other hand, there are also features which we need the information of previous expansions, e.g. the language models and the distortion penalty. Hence, each node additionally stores the target n-gram history $\tilde{e}$ and the end position j of the preceding source phrase. For a given coverage set O, there are various $\tilde{e}$ and j, depending on the expansion sequence. Thus, each node is labeled with a triple $(O, \tilde{e}, j)$. Figure 2.4 shows an example of such a search

graph. The decoding problem is equal to finding the optimal path in a search graph.

Note that the target sentence is generated in monotonous order, i.e. target positions are covered from 0 to I, consecutively. On the other hand, there is no specific order of the source positions when choosing source phrases for expansion. This makes the implementation simpler while allowing arbitrary phrase reorderings. In addition, we can make use of n-gram history on the target side even beyond the phrase boundaries.

2.5.3 Pruning

The complexity of a search graph is exponential in the length of the given source sentence. Finding the optimal path in the graph is an NP-hard problem [Knight 99]. Hence, an approximate method is used to reduce the search space and speed up the translation.

Beam search [Jelinek 98] is the technique which most phrase-based decoders adopt for this purpose. The idea is to prune unreliable hypotheses in the middle of traversing a search graph, according to the partial score for the traversed path. Additionally, we also consider the score for the remaining path to the end node, which cannot be exactly computed but can be estimated with heuristics. This is denoted as rest cost estimation, for which we apply the heuristics of [Zens 08] in this work. It intends to protect those hypotheses from being pruned, which have low partial scores at early stage but might be expanded to promising translations at completion. The degree of pruning is controlled by two kinds of parameters: The histogram size N_h is the number of hypotheses not to be discarded [Steinbiss & Tran$^+$ 94], and threshold τ is the maximum score difference allowed between a hypothesis and the best hypothesis.

Pruning can be done at several levels in search. First, before the actual search starts, we limit the number of target phrases per given source phrase. Here, we hope to filter out meaningless phrase pairs extracted from wrong word alignments. This pruning is based on the standard scores in the phrase table and language model score within the target phrase boundaries. Secondly, after we construct all the hypotheses which share the same coverage set O, we perform pruning among them. In the third level, we consider all the hypotheses with a given cardinality c, which is the number of source positions covered by O. Finally, for each cardinality c, we ignore those coverage sets whose the most probable hypothesis accords with pruning criteria.

For fair comparison of costs with respect to covered source positions, we traverse a search graph according to the cardinality of coverage sets. From the root node, we first construct all hypotheses of cardinality $c = 1$, using only single-word phrase translation rules. Then we proceed to cardinality $c = 2$ by expanding hypotheses of cardinality 1 with single-word phrase pairs, or the empty hypothesis with two-word

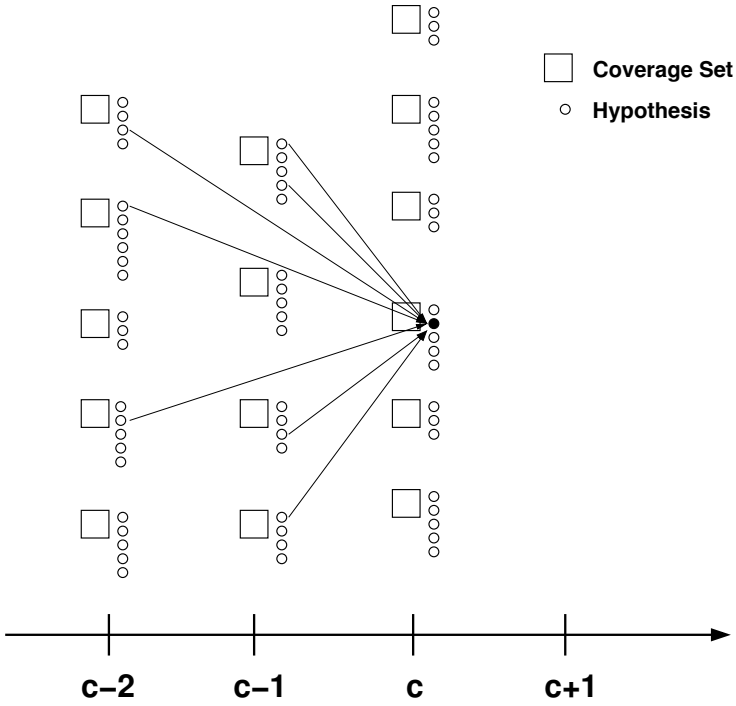


Figure 2.5: [Zens 08] Illustration of the source cardinality synchronous search. For each cardinality, we have a list of coverage sets (boxes). For each coverage set, we have a list of hypotheses (circles). A hypothesis of cardinality **c** (filled circle) can be expanded from a hypothesis of cardinality **c-1** with a single-word phrase, or from a hypothesis of cardinality **c-2** with a two-word phrase, and so forth.

phrase pairs. The search is continued in this way until we reach the maximum cardinality of coverage sets, i.e. the length of given source sentence. This search scheme is called source cardinality synchronous search, which preserves a topological order in traversing a search graph. Figure 2.5 illustrates the search procedure. The algorithm can be found in [Zens 08].

Chapter 3

Word Classes

In this work, we adopt word classes to refine several components of phrase-based SMT. This chapter reviews the definition, properties and estimation methods of word classes.

The concept of automatically generated word classes first appears in [Brown & deSouza$^+$ 92] for smoothing language models. They perform a clustering over words based on their collocations in the given corpus without any linguistic knowledge. The output is a set of classes, each of which contains words of similar meanings or syntactic roles. Table 3.1 shows some examples of the output.

Class 1:	`had hadn't would've could've should've must've might've`
Class 2:	`head body hands eyes voice arm seat eye hair mouth`

Table 3.1: [Brown & deSouza$^+$ 92] Examples of 1000 classes from a 260,741-word English vocabulary.

Class 1 can be seen as a syntactic class. Its members are abbreviated forms (except `had`) of auxiliary verb phrases expressing a situation of the past. They are gathered in the same class because they appear after a subject and are followed by a verb in past perfect tense. POS tags categorize words in a similar manner, but using a classification model. Most POS tagging models should be trained differently for language with human-annotated data, which is very expensive. Even for the same language, the predefined tagsets should agree for different datasets. On the contrary, the estimation of word classes requires no such pre-training and is universal for all languages. Furthermore, word classes can be a more fine-grained grouping than POS tags, if the number of classes is set relatively large. For example, a standard POS tagset classifies English verbs according to only conjugation (Table 3.2) while Class 1 captures writing style and tense concurrently.

Class 2 can be seen as a semantic class about the human body. The words of Class 2 can be located along with the same verb, e.g. "`move`". Note that "`seat`" is not a body part but closely related to a body pose. Such semantics is based on the *distributional* hypothesis, stating that words occurring in the same context tend to

VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present

Table 3.2: POS tags used for verbs in the Penn TreeBank English corpus [Marcus & Marcinkiewicz$^+$ 93].

purport similar meanings [Harris 54]. In computational linguistics, distributional semantics can be modeled with spectral methods [Deerwester & Dumais$^+$ 90,Dhillon & Foster$^+$ 11], topic models [Dinu & Laputa 10] or neural networks [Huang & Socher$^+$ 12, Mikolov & Sutskever$^+$ 13]. With proper similarity measures, these semantic representations can be grouped into semantic classes using traditional clustering algorithms, e.g. k-means or nearest neighbors [Schütze 98]. Compared to the word classes we use in this work, they require a more complex implementation and much longer estimation time, while the performance gain is marginal in various NLP applications [Turian & Ratinov$^+$ 10].

From our experience, automatically generated word classes are not well suited to group morphological variations of the same stem. One reason might be that a morphological inflection often changes the syntactic position of the word in a sentence, making the collocation statistically different.

Here, we introduce our notations for word classes. We define a mapping $\mathcal{C}$ from word vocabulary $\mathbb{W}_e$ to class label set $\mathbb{C}$ as

$$\mathcal{C} : \mathbb{W}_e \rightarrow \mathbb{C}. \tag{3.1}$$

Each word $e \in \mathbb{W}_e$ is mapped to the corresponding class label $\mathcal{C}(e) \in \mathbb{C}$:

$$e \longmapsto \mathcal{C}(e) \tag{3.2}$$

$$e_1^I \longmapsto \mathcal{C}(e_1^I) = \mathcal{C}(e_1),\ ...,\ \mathcal{C}(e_I) \tag{3.3}$$

where the second line shows a mapping from a sequence of words to a sequence of classes.

Note that we assume a hard class assignment: Each word belongs to exactly one class, i.e. $p(C|e) = 1$ for $C = \mathcal{C}(e)$ and $p(C|e) = 0$ for $C \neq \mathcal{C}(e)$. Probabilistic (soft) classes, i.e. a word can be member of more than one class, have been used in several monolingual NLP tasks [Merialdo 94, Goldwater & Griffiths 07, Brody & Laputa 09, Chrupała 11]. When used in SMT, however, they significantly increase the complexity of modeling and decoding [Koehn & Hoang 07]. This is

because translation involves a conversion between two languages, which propagates the complexity blow-up from the source side to the target side. The hard class assumption avoids the expansion from a single word to multiple classes, facilitating the integration of word classes into phrase-based SMT with less complexity and an easier implementation [Wuebker & Peitz$^+$ 13].

For the application in translation, we denote the class mapping on the source side as $\mathcal{C}_f$ and on the target side as $\mathcal{C}_e$:

$$f \longmapsto \mathcal{C}_f(f) \tag{3.4}$$
$$e \longmapsto \mathcal{C}_e(e) \tag{3.5}$$

We denote the word-class mappings for phrases as follows:

$$\tilde{f}_k \longmapsto \mathcal{C}_f(\tilde{f}_k) = \mathcal{C}_f(f_{b_k}),\ ...,\ \mathcal{C}_f(f_{j_k}) \tag{3.6}$$
$$\tilde{e}_k \longmapsto \mathcal{C}_e(\tilde{e}_k) = \mathcal{C}_e(e_{i_{k-1}+1}),\ ...,\ \mathcal{C}_e(e_{i_k}) \tag{3.7}$$

3.1 Monolingual Clustering

To estimate a monolingual word-class mapping, a monolingual corpus $\{e_1^I\}$ is needed. The likelihood of the given corpus is defined with respect to a word-class mapping $\mathcal{C}$

$$\mathcal{L}_{\text{mono}}(\{e_1^I\}, \mathcal{C}) := \sum_{e_1^I} \log\ p(e_1^I|\mathcal{C}), \tag{3.8}$$

with the following objective function for each sentence e_1^I [Brown & deSouza$^+$ 92]:

$$p(e_1^I|\mathcal{C}) := \prod_{i=1}^{I} \Big[\underbrace{p(\mathcal{C}(e_i)|\mathcal{C}(e_{i-n+1}^{i-1}))}_{(3.9.1)} \cdot \underbrace{p(e_i|\mathcal{C}(e_i))}_{(3.9.2)} \Big]. \tag{3.9}$$

An optimal word-class mapping $\widehat{\mathcal{C}}$ is determined by the maximum likelihood estimation:

$$\widehat{\mathcal{C}} = \underset{\mathcal{C}}{\text{argmax}}\ \mathcal{L}_{\text{mon}}(\{e_1^I\}, \mathcal{C}) \tag{3.10}$$

Equation (3.9) is a reformulation of a typical word n-gram model. For each word position i, the equation is factorized into a class n-gram (3.9.1) and a class membership probability (3.9.2). The former guides $\widehat{\mathcal{C}}$ to have collocation preferences of class sequences, e.g. Class 1 of Table 3.1 should succeed a class of pronouns or person names. The latter enforces the distribution of words among classes,

preventing the optimization from assigning all words to one class. Both probabilities are estimated by relative frequencies

$$p(C|\widetilde{C}) = \frac{N(C,\widetilde{C})}{N(\widetilde{C})}, \tag{3.11}$$

$$p(e|C) = \frac{N(e)}{N(C)}, \tag{3.12}$$

where the counts of classes are defined as sums of word-level counts:

$$N(C) = \sum_{e:\mathcal{C}(e)=C} N(e), \tag{3.13}$$

$$N(C,\widetilde{C}) = \sum_{\substack{e:\mathcal{C}(e)=C \\ \widetilde{e}:\mathcal{C}(\widetilde{e})=\widetilde{C}}} N(e,\widetilde{e}). \tag{3.14}$$

for $e \in \mathbb{W}_e$, $C \in \mathbb{C}$ and $\widetilde{C} \in \mathbb{C}^{n-1}$. [Brown & deSouza$^+$ 92] suggest a bigram formulation, i.e. $n = 2$, which is the simplest case of contextual clustering yet gives substantial performance.

The exact optimization of $\mathcal{L}_{\text{mono}}$ is computationally infeasible. Instead, greedy algorithms can be used to find a reasonable local optimum. [Brown & deSouza$^+$ 92] apply a bottom-up hierarchical clustering. Starting from each word being its own class, a pair of classes is merged one at a time until the desired number of classes is reached. [Kneser & Ney 93] adopts the *exchange algorithm*, which loops over all words per iteration, tentatively moving each word to every class and assigning it to the class that most increases the likelihood (Algorithm 1). The algorithm stops with an arbitrary convergence criterion, which is normally defined by a maximum number of iterations.

The exchange algorithm empirically performs equally good as the bottom-up clustering with regard to the optimized likelihood value, but the computation is more efficient [Martin & Liermann$^+$ 98,Emami & Jelinek 05]. Besides, the bottom-up method still needs to run the exchange algorithm on its output classes to achieve its best performance [Brown & deSouza$^+$ 92]. Hence, we decide to use the exchange algorithm to estimate monolingual word classes in our experiments.

The most well-known implementation of the exchange algorithm is [Och 95], which is part of the GIZA++ toolkit [Och & Ney 03]. It integrates generic combinatorial optimization algorithms, e.g. simulated annealing or threshold accepting, into the exchange algorithm iterations to avoid bad local optima. [Martin & Liermann$^+$ 98] describe efficient strategies for updating the count statistics and computing the likelihood difference in the algorithm. [Botros 15] introduce a parallelized version of [Martin & Liermann$^+$ 98], which linearly accelerates the loop over

Algorithm 1 Exchange algorithm for monolingual clustering

Input: corpus $\{e_1^I\}$, number of classes K
Output: word-class mapping $\mathcal{C}$
  initialize $\mathcal{C}$
  repeat
    for all $e \in \mathbb{W}_e$ **do**
      for all $C \in \mathbb{C}$ **do**
        compute $\mathcal{L}_{\text{mono}}$ when e moves to C
      end for
      move e to the class with the largest increase in $\mathcal{L}_{\text{mono}}$
    end for
  until convergence condition is met
  return $\mathcal{C}$

classes (Line 4 of Algorithm 1). It is also equipped with the threshold accepting strategy, which is shown to perform best in [Och 95]. We adopt the implementation of [Botros 15] and add various initialization methods to it.

Our implementation has three adjustable parameters: #iter, ν, and γ. The algorithm iterations are divided into two phases; the threshold accepting phase and the hill climbing phase. #iter is the number of iterations in the hill climbing phase. ν determines the length of the threshold accepting phase: $\nu \times$ #iter is the number of iterations for the threshold accepting. γ controls how much worse likelihood values are allowed in the threshold accepting phase.

One can extend the algorithm by class trigrams, but it tends to overfit to training data [Botros 15]. Therefore, we stick to the bigram clustering in all of our experiments.

3.2 Bilingual Clustering

For the usages in SMT, word classes are estimated for both source language and target language. If we perform monolingual clustering on each language independently, however, we cannot expect the two sets of classes to correspond to each other. That is, a class representing a certain meaning may exist only on the source side or only on the target side. In this case, a class-to-class translation cannot be accurately modeled.

To increase the correlation between source classes and target classes, [Och 99] propose a bilingual clustering algorithm which exploits the bilingual corpus with its word alignments. They define a bilingual joint objective function for a given

bilingual corpus $\{(f_1^J, e_1^I)\}$:

$$\mathcal{L}_{\text{bi}}(\{(f_1^J, e_1^I)\}, \mathcal{C}_f, \mathcal{C}_e) := \sum_{(f_1^J, e_1^I)} \log p(f_1^J, e_1^I | \mathcal{C}_f, \mathcal{C}_e) \tag{3.15}$$

$$= \sum_{(f_1^J, e_1^I)} \log \Big\{ \underbrace{p(e_1^I|\mathcal{C}_e)}_{(3.16.1)} \cdot \underbrace{p(f_1^J|e_1^I; \mathcal{C}_f, \mathcal{C}_e)}_{(3.16.2)} \Big\} \tag{3.16}$$

where the translation probability (3.16.2) is modeled as

$$p(f_1^J|e_1^I; \mathcal{C}_f, \mathcal{C}_e) = \prod_{j=1}^{J} \left\{ \left[\prod_{i \in a_j} \underbrace{p(\mathcal{C}_f(f_j)|\mathcal{C}_e(e_i))}_{(3.17.1)} \right] \cdot \underbrace{p(f_j|\mathcal{C}_f(f_j))}_{(3.17.2)} \right\}, \tag{3.17}$$

which includes the class transition probability (3.17.1) and the class membership probability of a source word (3.17.2). As the monolingual case, relative frequencies are used to estimate both probabilities. $\mathcal{L}_{\text{bi}}$ has a prior only for target sentences (3.16.1), which is identically formulated as Equation (3.9). By performing the maximum likelihood estimation, one should obtain optimal word-class mappings for the source language ($\widehat{\mathcal{C}_f}$) and for the target language ($\widehat{\mathcal{C}_e}$) simultaneously:

$$(\widehat{\mathcal{C}_f}, \widehat{\mathcal{C}_e}) = \underset{\mathcal{C}_f, \mathcal{C}_e}{\text{argmax}} \ \mathcal{L}_{\text{bil}}(\{(f_1^J, e_1^I)\}, \mathcal{C}_f, \mathcal{C}_e) \tag{3.18}$$

As an alternative, one can estimate the target classes $\widehat{\mathcal{C}_e}$ first and the source classes $\widehat{\mathcal{C}_f}$ thereafter with $\widehat{\mathcal{C}_e}$ unchanged. Based on the fact that the prior (3.16.1) can be linearly separated from (3.16.2), the following two-step optimization is possible:

$$\widehat{\mathcal{C}_e} = \underset{\mathcal{C}_e}{\text{argmax}} \ \mathcal{L}_{\text{mono}}(\{e_1^I\}, \mathcal{C}_e) \tag{3.19}$$

$$\widehat{\mathcal{C}_f} = \underset{\mathcal{C}_f}{\text{argmax}} \ \mathcal{L}_{\text{bi}}(\{(f_1^J, e_1^I)\}, \mathcal{C}_f, \widehat{\mathcal{C}_e}) \tag{3.20}$$

This aims to make $\widehat{\mathcal{C}_f}$ consistent with monolingually well-formed $\widehat{\mathcal{C}_e}$. Compared to the two-step approach, we denote the Equation (3.18) as the one-step optimization.

[Och 99] modify the exchange algorithm (Algorithm 1) for the bilingual objective, shown in Algorithm 2. For the two-step approach, we run Algorithm 1 first and then Algorithm 2, skipping the loop of Line 3-8.

In [Och 99], the bilingual classes show superior performance to the monolingual classes in the alignment template approach [Och & Ney 04], which is the predecessor of current phrase-based SMT. Based on these results, there has been further research on jointly learning word classes of two languages. [Zhao & Xing$^+$

Algorithm 2 Exchange algorithm for bilingual clustering

Input: bilingual corpus $\{(f_1^J, e_1^I)\}$, number of classes (K_f, K_e)
Output: bilingual word-class mapping $(\mathcal{C}_f, \mathcal{C}_e)$

initialize $(\mathcal{C}_f, \mathcal{C}_e)$
repeat
    for all $e \in \mathbb{W}_e$ **do**
        for all $C_e \in \mathbb{C}_e$ **do**
            compute $\mathcal{L}_{\text{bil}}$ if e moves to C_e
        end for
        move e to the class with the largest increase in $\mathcal{L}_{\text{bil}}$
    end for
    for all $f \in \mathbb{W}_f$ **do**
        for all $C_f \in \mathbb{C}_f$ **do**
            compute $\mathcal{L}_{\text{bil}}$ if f moves to C_f
        end for
        move f to the class with the largest increase in $\mathcal{L}_{\text{bil}}$
    end for
until convergence condition is met
return $(\mathcal{C}_f, \mathcal{C}_e)$

05] propose bilingual spectral clustering and apply it to word alignment problem and phrase extraction. Their results are, however, limited to very small datasets and lack a comparison to monolingual classes. A bilingual clustering objective is also constructed with a projection scheme [Tackström & McDonald$^+$ 12] or an information theoretic approach [Faruqui & Dyer 13], which are tested only on NER tasks. To the best of our knowledge, we apply bilingual classes to a state-of-the-art phrase-based SMT system for the first time. We implement the one-step and two-step algorithms of [Och 99] on top of [Botros 15] and use them in the experiments.

Chapter 4

Word Classes in Phrase-based SMT

In the previous chapter, we saw that word classes effectively group related words and they can be efficiently estimated with the exchange algorithm. We now describe applications of word classes in the modeling, decoding, and word alignment problems of phrase-based SMT.

4.1 Word Class Models

Most of the models in phrase-based SMT are built upon a discrete space of words. For example, the n-gram language model (Equation (2.14)) is defined on $\mathbb{W}_e^n$, and the word-based lexicon model (Equation (2.13)) has an input space of $\mathbb{W}_e \times \mathbb{W}_f$. The parameters of these models are determined by word counts or word sequence counts. As mentioned in Section 2.3, these counts have severe sparsity problems due to the large size of a word vocabulary.

Word classes are an appropriate alternative for building more robust models, since their space $\mathbb{C}$ has a much smaller vocabulary while it still carries meaningful linguistic information of various types. As we saw in Equation (3.13) and Equation (3.14), class counts are computed by summing up counts of the words in the class; if a probability model is defined on a class vocabulary, class counts collect unstable word counts, e.g. ones, to make the distribution smoother. In formulating a model, we simply put word classes in place of the words assigned to them, obtaining the smoothed models. For phrase-based models, we do not need to trim or decompose the phrases—eventually rolling back to word-based models—for smoothing, retaining the level of phrases in the translation process.

Obviously, word class models cannot be more precise than the fine-grained word-level models. We can expect a performance gain when both types of models are interpolated. With the help of the log-linear model combination, word class models proposed here can be integrated as an additional model into the existing model combination (Equation 2.10). Additionally, we propose to linearly combine a word class model with an existing model. For instance, the phrase translation model can be interpolated with the smoothed word class model (denoted by the subscript

WC) as follows:

$$p_{\text{Phr}}(\tilde{f}|\tilde{e}) = \frac{\sigma \cdot p_{\text{WC}}(\tilde{f}|\tilde{e}) + N(\tilde{f}, \tilde{e})}{\sigma + N(\tilde{e})} \tag{4.1}$$

The interpolation maintains the underlying log-sum structure of Equation (2.11). The parameter σ controls the degree of smoothing.

4.1.1 Word Class Translation Model

For the phrase translation model, the most straightforward way of using word classes is to replace all words in the source and target phrases with their respective word classes. The model h_{wcPhr} has the same form as Equation (2.11) but employs the following probability instead of $p_{\text{Phr}}(\tilde{f}|\tilde{e})$:

$$p_{\text{wcPhr}}(\tilde{f}|\tilde{e}) = \frac{N(\mathcal{C}_f(\tilde{f}), \mathcal{C}_e(\tilde{e}))}{N(\mathcal{C}_e(\tilde{e}))} \tag{4.2}$$

The same approach can be applied to the word-based lexicon model. The model h_{wcLex} is basically defined as Equation (2.13), substituting $p(f|e)$ with the class transition probability:

$$p_{\text{wcLex}}(f|e) = \frac{N(\mathcal{C}_f(f), \mathcal{C}_e(e))}{N(\mathcal{C}_e(e))} \tag{4.3}$$

These two models are referred to as the word class translation model (wcTM) [Wuebker & Peitz$^+$ 13].

The wcTM can be trained without changing an off-the-shelf procedure for phrase-based model training, because it has the same structure as standard translation models [Wuebker & Peitz$^+$ 13]. If the training data is preprocessed by replacing each word with its word class, the same training procedure produces a phrase table with wcTM scores in lieu of h_{Phr} and h_{Lex}. Here, the word alignment is learned before this replacement. Consequently, one shall have two phrase tables: One with the original model scores and the other with word class model scores. Sorting both tables by the word classes, it is possible to walk through corresponding entries of the two tables and augment the original table with wcTM scores.

However, the steps presented above necessitates doubles the training effort and afford an extra storage for the second phrase table, which proves to be inefficient for large-scale SMT tasks. Therefore, we directly integrate the training of wcTM into the existing phrase-based training procedure, which is more efficient in terms of memory and time. Here, we present the core steps of the implementation:

1. Load word-class mappings $(\mathcal{C}_f, \mathcal{C}_e)$ into memory

2. When computing the word counts (e.g. $N(\tilde{f}, \tilde{e})$), dynamically map the words to word classes and increment also the counts of classes (e.g. $N(\mathcal{C}_f(\tilde{f}), \mathcal{C}_e(\tilde{e}))$)
3. Compute the standard translation model scores and the wcTM scores
4. Output all scores into one phrase table

This requires only one training pipeline, and is convenient in practice.

For a more rigorous modeling, we refine the phrase translation model of wcTM with class membership probability:

$$h_{\text{wcPhr+mbs}}(e_1^I, s_1^K; f_1^J) = \sum_{k=1}^{K} \log \left\{ p_{\text{wcPhr}}(\tilde{f}|\tilde{e}) \cdot \prod_{j=b_k}^{j_k} p(f_j|\mathcal{C}_f(f_j)) \right\} \tag{4.4}$$

The inverse model is defined in a similar manner with $p(e|\mathcal{C}_e(e))$. This refinement makes the model closer to the exact factorization of the standard phrase translation model. Because the class membership probability is higher for more frequent words in a class, this might filter out unlikely phrase translation rules containing less frequent words. Our word clustering software is able to generate the class membership probability table, which can be passed as an additional parameter to the wcTM training procedure.

4.1.2 Class Smoothing Model

Motivation

The wcTM generalizes all words of a phrase without distinction between them. This might cause the side effect that an absurd phrase gets a high score or a common phrase gets a low score. To illustrate this, we consider the English phrase "`move the voice`". It has a very limited use (only in music theory), so it should not be highly scored. We can expect that the clustering algorithm shown in Chapter 3 will assign "`move`" to a class of verbs about movement, "`the`" to a class of articles, and "`voice`" to a class about human body (see Class 2 of Table 3.1). wcTM replaces all words in the phrase with their corresponding word classes, generating a word class sequence which means "*move the body_part*" (generalized meanings are slanted). It is a plausible word class phrase and has a high count, thanks to frequent phrases such as "`move the body`", "`move the hands`, or "`move the arm`". Thus, the wcTM overrates the probability of the phrase "`move the voice`". This is because the word class mapping is not applicable to all kinds of contexts; the word "`voice`" is semantically close to the other words in the body part class, yet has a different usage alongside verbs.

The problem can be partially solved by generalizing only a sub-part of the phrase. When "`move`" is substituted with its corresponding word class and the other words

remain as they are, we obtain a phrase which is supposed to mean "*move* `the voice`". Since none of the movement verbs are compatible with `voice`, it should have a low count. In this manner, we are able to smooth the phrase counts without distorting the original meaning.

Model

This motivates us to develop the *class smoothing model* (CSM), which we replace one word at a time with the respective word class in the phrase translation model. To formulate CSM, we first introduce the notation for the selective class mapping in a word sequence:

$$\mathcal{C}^{(\{i\})}(e_1^I) := w_1,\ ...,\ \mathcal{C}(e_i),\ ...,\ e_I \tag{4.5}$$

where a set of word positions ($\{i\}$) in the superscript indicates which words are to be mapped to their word class. CSM calculates the average of the smoothed phrase translation probabilities for all replaced words within a phrase:

$$p_{\text{csm:src}}(\tilde{f}|\tilde{e}) = \sum_{j=b_k}^{j_k} \frac{w_j}{\sum_{j'} w_{j'}} \cdot p(\mathcal{C}_f^{(\{j\})}(\tilde{f})|\tilde{e}) \tag{4.6}$$

where w_j is the averaging weight. If we use uniform weights for all word positions j, the model computes the arithmetic mean. For example, the expansion of Equation (4.6) with a three-word phrase pair and $w_j = 1$ is:

$$\begin{aligned} p_{\text{csm:src}}(f_1\ f_2\ f_3 \mid e_1\ e_2\ e_3) = \frac{1}{3} \cdot \Big[& p(\mathcal{C}_f(f_1)\ f_2\ f_3 \mid e_1\ e_2\ e_3) \\ & + p(f_1\ \mathcal{C}_f(f_2)\ f_3 \mid e_1\ e_2\ e_3) \\ & + p(f_1\ f_2\ \mathcal{C}_f(f_3) \mid e_1\ e_2\ e_3) \Big] \end{aligned} \tag{4.7}$$

The probabilities of Equation (4.6) are tallied up in a log-linear way as Equation (2.11). Initially, we back off only source words, thus this model is referred to the CSM source ($h_{\text{csm:src}}$). We also estimate the inverse model with $p_{\text{csm:src}}(\tilde{e}|\tilde{f})$, for which the source phrases in the conditioning part are generalized.

Next, we define the CSM source + target ($h_{\text{csm:src+tgt}}$), which generalizes also target words:

$$p_{\text{csm:src+tgt}}(\tilde{f}|\tilde{e}) = \sum_{j=b_k}^{j_k} \frac{w_j}{\sum_j w_j} \cdot p(\mathcal{C}_f^{(\{j\})}(\tilde{f})|\mathcal{C}_f^{(a_j)}(\tilde{e})) \tag{4.8}$$

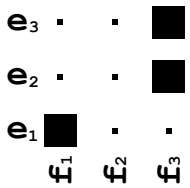


Figure 4.1: Word alignments of the three-word phrase pair in Equation 4.9.

For each source position j, we replace also the target words aligned to the source word f_j. The model therefore implicitly encapsulates the alignment information. The following equation is an example:

$$\begin{aligned} p_{\text{csm:src+tgt}}(f_1\ f_2\ f_3 \mid e_1\ e_2\ e_3) = \frac{1}{3} \cdot \Big[& p(\mathcal{C}_f(f_1)\ f_2\ f_3 \mid \mathcal{C}_e(e_1)\ e_2\ e_3) \\ & + p(f_1\ \mathcal{C}_f(f_2)\ f_3 \mid e_1\ e_2\ e_3) \\ & + p(f_1\ f_2\ \mathcal{C}_f(f_3) \mid e_1\ \mathcal{C}_e(e_2)\ \mathcal{C}_e(e_3)) \Big] \end{aligned} \tag{4.9}$$

where the word alignments are depicted with line segments and the target word classes are colored in red. Figure 4.1 shows the alignment diagram for this phrase pair. f_1 is aligned to e_1, which is backed off to its target class. As f_2 has no alignment points, we do not replace any target word accordingly. f_3 triggers the class replacement of two target words at the same time.

The two types of CSM can be also formulated in the reverse direction; we back off the target words first and the aligned source words subsequently. This leads to another pair of models: CSM target ($h_{\text{csm:tgt}}$) and CSM target + source ($h_{\text{csm:tgt+src}}$).

Additionally, we can revise the CSM with the class membership probability as wcTM (here for $h_{\text{csm:src}}$):

$$p_{\text{csm:src}}(\tilde{f}|\tilde{e}) = \sum_{j=b_k}^{j_k} \frac{w_j}{\sum_j w_j} \cdot p(f_j|\mathcal{C}_f(f_j)) \cdot p(\mathcal{C}_f^{(\{j\})}(\tilde{f})|\tilde{e}) \tag{4.10}$$

The CSM constructs phrase pairs with both words and word classes. They are more robust to unwanted elevations or degradations of translation scores, which is described at the beginning of this subsection. Furthermore, we can directly utilize these phrases for a novel paraphrasing method in decoding (Section 4.2).

Weighting Schemes

Depending on how we set the averaging weight, the CSM can have different scoring schemes on the word level. We present three alternatives of weighting for

CSM source direction models, which can be similarly applied to the reverse models. Firstly, we use inverse unigram of the replaced source word:

$$\frac{1}{w_j} = \frac{N(f_j)}{\sum_{f'} N(f')} \tag{4.11}$$

which puts more weight on backing off rare words. The intuition here is that a rare word is the main reason for unstable counts and should be smoothed above all. The second alternative is an extension of the previous idea to the phrase level:

$$\frac{1}{w_j} = \frac{N(f_{b_k} \ldots f_j \ldots f_{j_k})}{\sum_{f'} N(f_{b_k} \ldots f' \ldots f_{j_k})} \tag{4.12}$$

which is the probability of f_j being in the current source phrase among all possible source words. We call this scheme the inverse source phrase replacement probability. Lastly, we introduce factorizing likelihood:

$$w_j = N(\mathcal{C}_f^{(\{j\})}(\tilde{f})) \tag{4.13}$$

which is the count of the generalized phrase. It plays a similar role as the previous two forms, since the count is higher when a rare word is replaced with a word class. From the implementation point of view, factorizing likelihood is simpler, because we make use of the marginal counts of the inverse model while the others need extra counting.

4.1.3 Word Class Language Model

For smoothing the target side language model, we build a word class language model h_{wcLM} which redefines the n-gram probability term of Equation (2.14) with word classes:

$$p(e_i|e_1^{i-1}; \mathcal{C}_e) = p(\mathcal{C}_e(e_i)|\mathcal{C}_e(e_{i-n+1}^{i-1})) \tag{4.14}$$

where all words are replaced with their respective classes in the formulation. It models n-gram contexts of word class sequences. Since the class vocabulary reduces the sparsity of the sequence counts, longer n-gram context can be modeled efficiently [Wuebker & Peitz$^+$ 13].

We can incorporate the class membership probability also in the wcLM:

$$p(e_i|e_1^{i-1}; \mathcal{C}_e) = p(e_i|\mathcal{C}_e(e_i)) \cdot p(\mathcal{C}_e(e_i)|\mathcal{C}_e(e_{i-n+1}^{i-1})) \tag{4.15}$$

This is the conventional formulation of word class n-gram models for language modeling [Brown & deSouza$^+$ 92], but not widely used in machine translation. In

Section 5.3, we empirically verify the utility of the class membership probability in wcLM for SMT.

Both forms of wcLM can be easily trained using an existing language model toolkit [Stolcke 02], if the monolingual training data is preprocessed in the same way as in Section 4.1.1.

4.1.4 Other models

Other types of models can be also candidates for word class smoothing. As an example, we apply the same idea of class replacements to hierarchical reordering model (HRM) [Galley & Manning 08], named as wcHRM [Wuebker & Peitz$^+$ 13].

4.2 Word Class Decoding

Motivation

We cannot extract all possible phrase pairs from a limited amount of bilingual data, resulting in a specific upper limit for the improvement with better modeling of the existing phrase pairs. This encourages the method of generating additional phrase pairs which supplements the conventional phrase extraction. One possible way is to manipulate the extracted phrases with linguistic variations.

Word classes can be used to replace each word in a phrase, since they have words which are linguistically linked to the replaced word. When a word in a phrase is substituted with its corresponding word class, we can expand the class to other member words within the same class, creating new phrase pairs which could not be extracted from the training corpus before.

If word classes are estimated on a large monolingual corpus, we might also have class mappings for OOV words. In such cases, we can obtain phrase pairs including OOV words. Since it is much easier to acquire monolingual data than bilingual data, it should be particularly attractive for low-resource SMT tasks.

We realize this idea in the standard phrase-based decoder to use those paraphrases as additional translation options, enlarging the search space.

Relation to Class Smoothing Model

To this end, we reuse the generalized phrase pairs for the CSM. They are originally the phrase pairs extracted from the training data, but some words are replaced with their word classes. We therefore only need to expand those classes for obtaining the paraphrases. Also, CSM phrase pairs restrict the class replacement to a single word on one side and the aligned words on the other side, which is a reasonable trade-off

between efficiency and diversity of the paraphrasing. If more words are changed, then we are abundant in paraphrases, however, their average quality becomes worse; too many substitutions on the word level may distort the meaning of the original phrase, where the local context is not guaranteed to be preserved. If fewer words are changed, then the whole procedure becomes efficient, but we may be short of useful paraphrases.

When used in the translation process, the source side of a CSM phrase pair does not require actual expansions of classes to words. For a phrase segment of the given source sentence, we are interested not in its paraphrases themselves, but in the target translations of the paraphrases (Figure 4.2). On the other hand, we need to expand the classes on the target side, producing word-only phrases which can be written in the translation output (Figure 4.3).

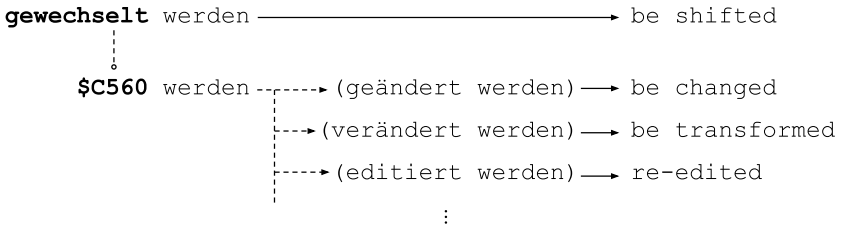


Figure 4.2: Paraphrasing of a German phrase using the CSM source model. A dashed edge with a circle tip represents a word-class mapping. Dashed arrows indicate class expansions. "$C" is the prefix for class labels. The German phrase "`gewechselt werden`" originally has only a translation to "`be shifted`", but can be translated to its synonyms, e.g. "`be changed`", by the paraphrasing. Phrases in parentheses are source paraphrases, which are not explicitly constructed in decoding. Extracted from IWSLT 2012 German→English TED talk data with 1000 monolingual classes on the source side.

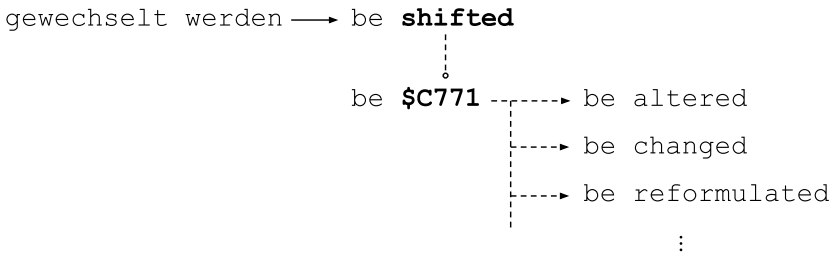


Figure 4.3: Paraphrasing of a German phrase using the CSM target model. Data source and legends are the same as Figure 4.2. The target phrase "`be shifted`" is rephrased to e.g. "`be altered`", by the class expansions. Used 1000 monolingual classes on the target side.

The standard phrase-based decoder can directly use the CSM phrase pairs as translation rules only if we compute all model scores for them. The source-to-target and target-to-source phrase translation scores are equal to the CSM scores, for which we can profit from the same training procedure of the CSM. For the word-based lexicon model scores, we compute class-to-word ($p(f|C_e)$, $p(e|C_f)$), word-to-class ($p(C_f|e)$, $p(C_e|f)$) and class-to-class ($p(C_f|C_e)$, $p(C_e|C_f)$) translation lexica. Whenever a class is located in place of a word in the CSM phrase pair, one of these probabilities stands in for the word translation lexicon. As an example, for the first CSM phrase pair in Figure 4.2, $p(\texttt{\$C560}|\texttt{be})$ and $p(\texttt{\$C560}|\texttt{changed})$ are used instead of $p(\texttt{gewechselt}|\texttt{be})$ and $p(\texttt{gewechselt}|\texttt{changed})$. These lexica can be easily computed using the existing word-based lexicon and a bilingual word-class mapping. Besides, the majority of the models can be scored by adding up the scores of all the paraphrases, e.g. the wcTM and the lexicalized reordering models (LRM). The models which do not consider lexical identities, e.g. penalty terms, do not necessitate special care.

A CSM phrase pair and its model scores constitute a complete phrase table entry. Such entries are appended to the original phrase table, queried by the decoder on an equal footing with the normal phrase pairs. It should be noted that CSM phrase pairs have a different scale of scores from the original phrase pairs. This is due to the aggregation of counts with respect to the classes. We resolve this discrepancy by introducing the membership probabilities of the class replacements. They have the effect of refactorizing the class-based formulation to be on the word space. We use the product of these probabilities as an additional model in the log-linear combination, rather than multiplying each model score by them:

$$h_{mbs}(e_1^I, s_1^K; f_1^J) = \sum_{k=1}^{K} \left[\sum_{j=b_k}^{j_k} \mathbf{1}_{B_f}(j) \cdot \log p(f_j|\mathcal{C}_f(f_j)) + \sum_{i=i_{k-1}+1}^{i_k} \mathbf{1}_{B_e}(i) \cdot \log p(e_i|\mathcal{C}_e(e_i)) \right] \tag{4.16}$$

where $\mathbf{1}_B$ is an indicator function with a set of back-off positions B. For a normal phrase pair without any class, this score is zero, which does not affect the total score. The advantage here is that the degree of the rescaling (λ_{csm}) can be automatically tuned by MERT, maximizing the translation performance on a development corpus.

Decoder Modifications

Algorithm 3 summarizes the necessary changes in the decoding algorithm for exploiting the CSM phrase pairs. The major change occurs in the phrase matching step (Section 2.5.1). For each source phrase in the input sentence, we also seek for

Algorithm 3 Phrase-based SMT decoding with CSM phrase pairs

Input: input sentence f_1^J, phrase table T extended with CSM phrase pairs
Output: target translation e_1^I

Variables:

S = a set of source positions for the class replacement
$matchList[\tilde{f}]$ = a list of target phrases matched to $\tilde{f}$
$matchListCSM[\tilde{f}][S]$ = a list of target phrases matched to $\mathcal{C}_f^S(\tilde{f})$
N_h = observation histogram size for $matchList[\tilde{f}]$
N_{csm} = observation histogram size for $matchListCSM[\tilde{f}][S]$

Phrase Matching:

for all $\tilde{f}$ **in** f_1^J **do**
  $matchList[\tilde{f}] \leftarrow T(\tilde{f})$
  for all $S \in \mathcal{P}(\{j \mid f_j \in \tilde{f},\ N(f_j) < \tau_u\}) \setminus \emptyset$ **do**
    for all $\tilde{e} \in T(\mathcal{C}_f^S(\tilde{f}))$ **do**
      for all C_e **in** $\tilde{e}$ **do**
        expand C_e to top N_t frequent member words of C_e
      end for
      $matchListCSM[\tilde{f}][S] \leftarrow$ all class expansions of $\tilde{e}$
    end for
  end for
end for

Search Graph:

for each next source phrase $\tilde{f}$ to be translated **do**
  expand hypotheses with best N_h translations in $matchList[\tilde{f}]$
  for all $l \in matchListCSM[\tilde{f}]$ **do**
    expand hypotheses with best N_{csm} translations in l
  end for
end for
$e_1^I \leftarrow$ optimal path of the search graph
return e_1^I

the matching CSM entries in the phrase table. The result of each query (a list of target translations) is denoted by$T(\tilde{f})$, where T is the phrase table and $\tilde{f}$ is the search key.

We limit the amount of CSM phrase queries with parameter τ_u, which indicates the maximum unigram count of those words to be substituted with their classes. The idea presumes that a frequent word does not need any generalization, since we

already have enough statistics for that word. We can concentrate on backing off only rare words by setting τ_u low.

The retrieved target phrases contain word classes that have to be expanded, except for CSM source. We define a parameter N_t as the number of the member words used to expand each class. The words are chosen by their relative frequency within the class in order to avoid the paraphrasing with unusual words. Once the classes are replaced with words, the decoder can use the target paraphrases as possible translation options; e.g. the language model scores can be computed for the hypothesis expansions.

We keep separate lists for the original phrase matches and the CSM phrase matches, and also apply different observation histogram parameters to both types of matching lists. This allows to control the effect of the paraphrasing. Other pruning details in search are omitted in Algorithm 3.

4.3 Word Class Alignments

Section 2.1 provides motivation for an additional word alignment which is different from the original word alignment. We introduce the word class alignment, which is the translation correspondence between source and target classes. Each word class represents a syntactic element or a semantic category, we therefore expect its alignments to be linguistically meaningful.

To learn word class alignments, we first replace every word in the training corpus with its word class. Afterwards, we train the statistical alignment models on the word class corpus, using GIZA++. The output alignment is used in the following two ways:

1. **Merge with the original word alignment.** It is a typical method to improve the overall quality of word alignments, aiming to synthesize different information in multiple word alignments. We use alignment merging heuristics introduced by [Och & Ney 00]. The merged alignment is used to extract phrase pairs and train all individual models.

2. **Extract phrase pairs and train word class models.** We can directly extract phrase pairs from word class alignments, hoping that their count statistics are better suited for the wcTM (Section 4.1.1) or the CSM (Section 4.1.2). These phrases are only used to score the wcTM or the CSM of the original phrase pairs, i.e. they are not used as additional phrase table entries.

   Note that the phrase pairs from word class alignments do not cover all of the original phrase pairs. For the phrase pairs extracted from the original word alignment, the wcTM or the CSM scores are computed based on the ori-

ginal phrase counts as before. We add an extra penalty cost to such entries, conversely promoting the models trained with word class alignments.

Chapter 5

Experiments

In this chapter, we present experimental results of our proposed methods: CSM, CSM paraphrasing, and word class alignments. We also empirically analyze the state-of-the-art word class models—wcTM, wcLM, and wcHRM—and their refinements. In addition, we investigate the effect of word class estimation to the performance of these methods. Before that, we describe the system, evaluation methodology, and datasets where the experiments are conducted.

5.1 Test Environment

System

We implement the approaches in Chapter 4 inside the open source phrase-based SMT toolkit Jane 2 [Wuebker & Huck$^+$ 12], and use it for all of the translation experiments.

As a baseline system, we train and integrate the following models into the log-linear combination:

- Phrase translation model (Section 2.3): both directions
- Word-based lexicon model (Section 2.3): both directions
- Word penalty
- Phrase length penalty
- Distortion penalty (Section 2.3)
- Target side language model (Section 2.3): 4-gram
- Target side word class language model (Section 4.1.3) 7-gram, 100 classes, without membership probability
- Hierarchical lexicalized reordering model [Galley & Manning 08]

The model weights are trained with MERT. Since MERT is a nondeterministic algorithm, we run it three times and choose the single set of lambdas which gives the best translation performance.

Evaluation

The translation quality is measured by two automatic evaluation metrics: BLEU [Papineni & Roukos$^+$ 02] and TER [Snover & Dorr$^+$ 06].

Datasets

We use German→English in-domain data of International Workshop on Spoken Language Translation (IWSLT) 2012 Evaluation Campaign MT track[1] [Cettolo & Girardi$^+$ 12] as the primary dataset for our experiments. It is extracted from TED conference talk subtitles, and has a relatively small size (about 2.5M running words on the training data). Table 5.1 shows the corpus statistics. We simulate low-resource SMT scenarios with this data, which we expect our methods to be especially useful.

We also test our methods on large-scale SMT tasks of Workshop on Statistical Machine Translation (WMT), whose training data is composed of Europarl corpus [Koehn 05], News Commentary corpus, and Common Crawl corpus. The first two data are formal speeches, and the last is a collection of random web texts. Table 5.2 provides the corpus statistics of WMT 2014 English→German data[2].

We use also WMT 2015 English→Czech[3] data, where the corpus statistics are shown in Table 5.3. Note that, in contrast to the IWSLT dataset, the morphologically richer language is on the target side in the WMT tasks. As mentioned in 2.3, rich morphology causes more sever sparsity problems. The models which generalize the target side, e.g. wcLM or CSM source + target, are expected to be more effective in these tasks.

5.2 Optimizing Word Classes

Our proposed methods are dependent on the given word class mapping, whose estimation varies with several factors, e.g. parameters of the clustering algorithm. Here, we perform a series of experiments to discover the relation between the word class mapping and the word class models trained with them. We ultimately intend to find the optimal class mapping which maximizes the performance of our methods.

As a first step, we optimize the word clustering without considering the purpose in translation. We tune the clustering parameters to produce good values of common clustering quality measures. Next, more importantly, we evaluate different word class mappings with respect to the actual translation quality when they are used in

[1] https://wit3.fbk.eu/mt.php?release=2012-03
[2] http://www.statmt.org/wmt14/translation-task.html
[3] http://www.statmt.org/wmt15/translation-task.html

		German	English
train	Sentences	130K	
	Running Words	2.5M	2.5M
	Vocabulary	71K	49K
dev[†]	Sentences	883	
	Running Words	20K	21K
	Vocabulary	4K	3K
	OOVs (Rate)	776 (4%)	639 (3%)
test	Sentences	1565	
	Running Words	32K	27K
	Vocabulary	5K	5K
	OOVs (Rate)	1068 (3%)	753 (2%)

Table 5.1: Corpus statistics of IWSLT 2012 German→English data.

		English	German
train	Sentences	4M	
	Running Words	104M	105M
	Vocabulary	648K	659K
news-test2011	Sentences	3003	
	Running Words	66K	81K
	Vocabulary	14K	13K
	OOVs (Rate)	2128 (3%)	1736 (2%)
news-test2012[†]	Sentences	3003	
	Running Words	73K	81K
	Vocabulary	10K	13K
	OOVs (Rate)	1827 (2%)	1688 (2%)
news-test2013	Sentences	3000	
	Running Words	56K	70K
	Vocabulary	13K	12K
	OOVs (Rate)	1426 (2%)	1310 (2%)

Table 5.2: Corpus statistics of WMT 2014 English→German data.

[†]The development set used for tuning the model weights.

		English	Czech
train	Sentences	930K	
	Running Words	2.4M	2.1M
	Vocabulary	161K	345K
news-test2012[†]	Sentences	3003	
	Running Words	73K	65K
	Vocabulary	10K	17K
	OOVs (Rate)	1336 (2%)	2393 (4%)
news-test2013	Sentences	3000	
	Running Words	65K	57K
	Vocabulary	9K	15K
	OOVs (Rate)	1170 (2%)	2023 (4%)
news-test2014	Sentences	3003	
	Running Words	69K	60K
	Vocabulary	9K	16K
	OOVs (Rate)	1298 (2%)	2190 (4%)

Table 5.3: Corpus statistics of WMT 2015 English→Czech data.

our proposed models. This reveals the adequacy of the clustering measures in the context of phrase-based SMT. We further examine how the following factors influence the translation quality: clustering iterations, initialization of the clustering, the number of classes, and the clustering algorithm.

5.2.1 Clustering Measures

To assess the quality of a clustering, the most intuitive measure is the clustering objective function itself. The value indicates how well the output word class mapping is optimized as we have intended. For the monolingual clustering, it is basically an alternative n-gram model of the given corpus (Equation (3.8–3.9)). In this case, we can transform the objective to *perplexity*:

$$PPL(\{e_1^I\}) = \Big[\prod_{e_1^I} p(e_1^I)\Big]^{-\frac{1}{\sum_{e_1^I} I}} \tag{5.1}$$

$$= \exp\left(-\frac{1}{\sum_{e_1^I} I}\sum_{e_1^I}\log p(e_1^I)\right) \tag{5.2}$$

[†]The development set used for tuning the model weights.

where Equation (3.8) is a part of it. Perplexity can be interpreted as the number of choices per word position. The lower value is preferred, which means the model has less uncertainty for the given corpus. Since it is widely accepted as a standard measure in NLP, we use perplexity to measure the quality of monolingual clusterings.

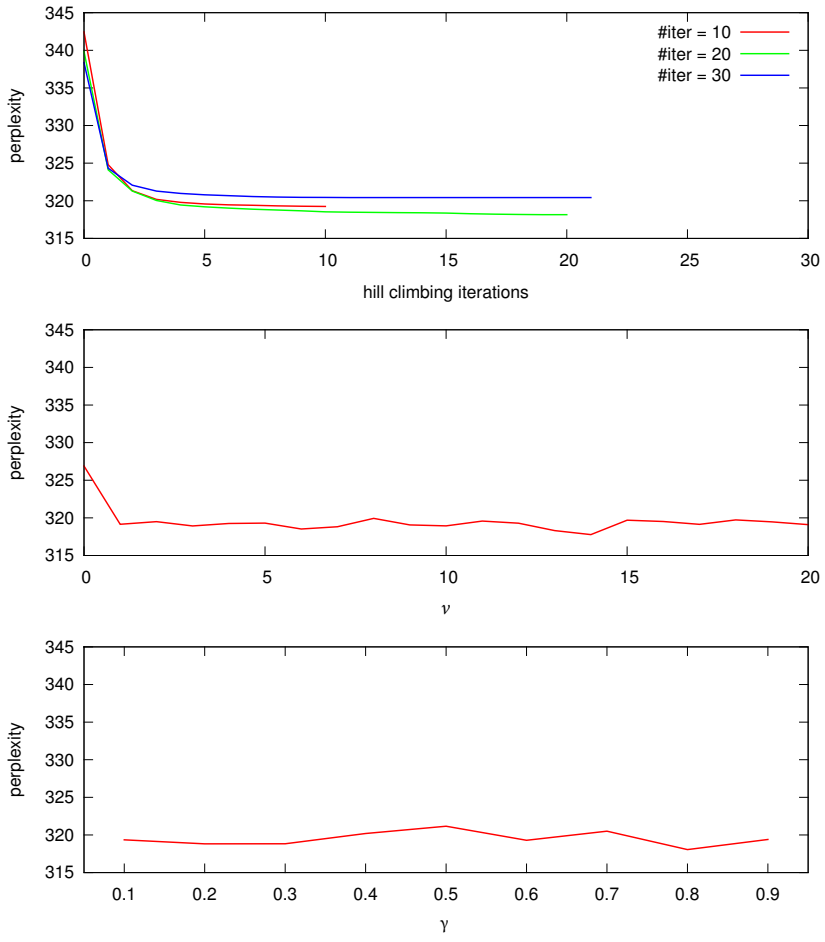


Figure 5.1: Perplexity curves over hill climbing phase iterations (top), ν (middle), and γ (bottom) for monolingual clustering on the source side of WMT 2014 English→German data. The top figure has individual curves for different values of #iter.

The exchange algorithms for word clustering have three parameters: #iter, ν, and γ. Figure 5.1 shows the perplexity curves for tuning these parameters. For three values of #iter (10, 20, 30), the difference in the final perplexity values is negligible (around 1–2). Moreover, even if we set #iter = 30, the algorithm already

converges at hill climbing iteration 21. We decide to use 10 for #iter, because we cannot expect further meaningful change of the perplexity with a higher value. The curves over ν and γ are nearly flat: the parameters seem not to affect the clustering quality much. We choose $\nu = 2$, and $\gamma = 0.4$, which are empirically acceptable values from an extensive study on these parameters in [Och 95].

For bilingual clustering, it is impossible to use the perplexity criterion to evaluate the word class mapping. The bilingual objective function covers the probability of two languages, which does not fit to the definition of perplexity (Equation (5.2)). Therefore, we directly use the log-likelihood value (Equation (3.15)) instead. We put the negative sign in front of it for the consistency in the score comparison: the lower is better.

We introduce other two metrics from previous literature on bilingual clustering. Class transition perplexity [Och 99] formulates the translation probability of bilingual sentences in a similar way as perplexity:

$$CTPPL(\{(f_1^J, e_1^I)\}) = \left[\prod_{(f_1^J, e_1^I)} p(f_1^J | e_1^I) \right]^{-\frac{1}{\sum_{f_1^J} J}} \tag{5.3}$$

$$= \exp\left(-\frac{1}{\sum_{f_1^J} J} \sum_{(f_1^J, e_1^I)} \log p(f_1^J | e_1^I) \right) \tag{5.4}$$

$$= \exp\left(-\frac{1}{\sum_{f_1^J} J} \sum_{(f_1^J, e_1^I)} \sum_{j=1}^{J} \log \max_e p(\mathcal{C}_f(f_j) | \mathcal{C}_e(e)) \right) \tag{5.5}$$

where the translation probability $p(f_1^J | e_1^I)$ is modeled with the maximum probable class transition to each source class, which is much simpler than Equation (3.17). Average ϵ-mirror size [Wang & Lafferty$^+$ 96] is defined not on the bilingual corpus but on the class mappings themselves:

$$\epsilon MIRROR_{avg}(\mathcal{C}_f, \mathcal{C}_e) = \frac{1}{K_e} \sum_{C_e \in \mathcal{C}_e} \left| \{ C_f \mid C_f \in \mathcal{C}_f,\ p(C_f \mid C_e) > \epsilon \} \right| \tag{5.6}$$

The ϵ-mirror of a target class is the number of source classes which have the translation probability greater than ϵ. We set $\epsilon = 0.05$ by default in the experiments, following [Och 99]. Both measures gauge the variance of the translation probability for each class. A small value means that the class translations are very focused and that there is less ambiguity in predicting the target class given a source class.

The clustering measure curves for parameter tuning of bilingual one-step clustering are given in Figure 5.2. The tendency of the curves is the same as the monolingual case. #iter larger than 10 does not provide a noticeable improvement

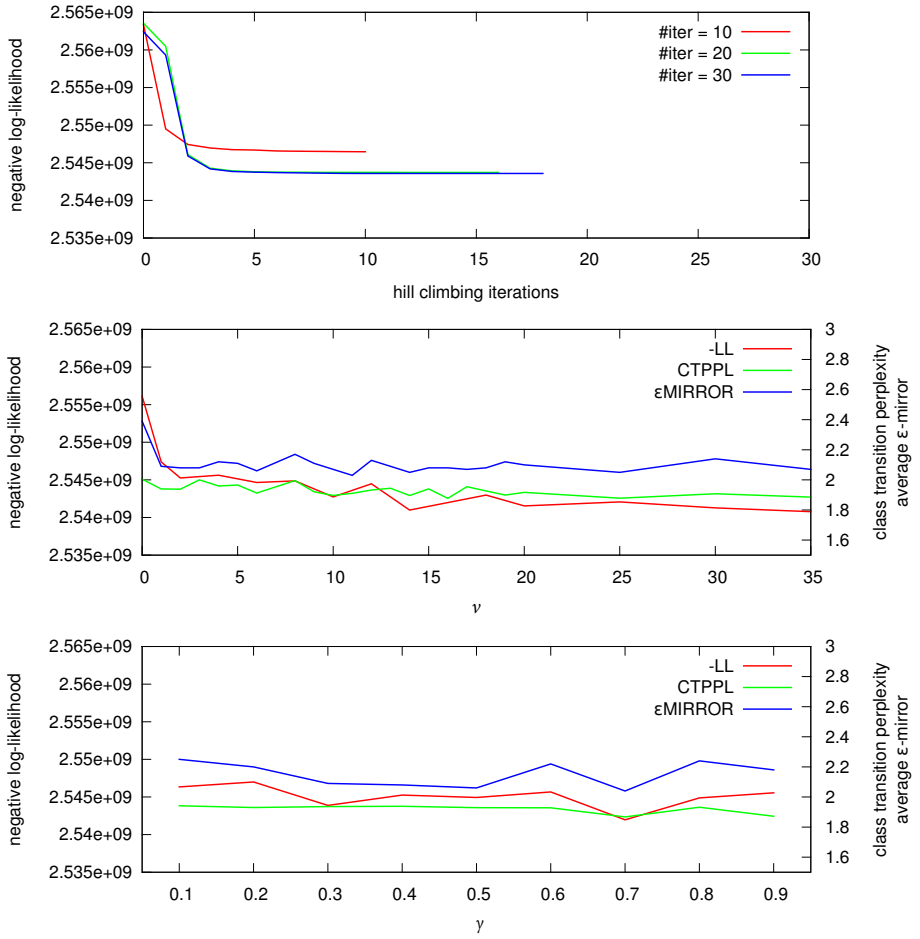


Figure 5.2: Clustering measure curves over hill climbing phase iterations (top), ν (middle), and γ (bottom) for bilingual one-step clustering on WMT 2014 English→German data. The top figure has negative log-likelihood curves for different values of #iter. The middle and bottom figures have individual curves for different clustering measures.

in negative log-likelihood. There are no clear optima in the curves over ν and γ. We use the same default values as the monolingual clustering for these parameters. We have similar results for the bilingual two-step clustering (see Figure A.1).

The parameters are so far tuned on the given training corpus, while the actual translation is conducted on other corpora. Suspecting that this might cause overfitting of the word class estimation to the training data, we calculate the clustering measures also on the development and test data. Figure 5.3 (for monolingual clustering) and 5.4 (for bilingual one-step clustering) show the measure values over the clustering iterations. We can see that the values are changing over the iterations in

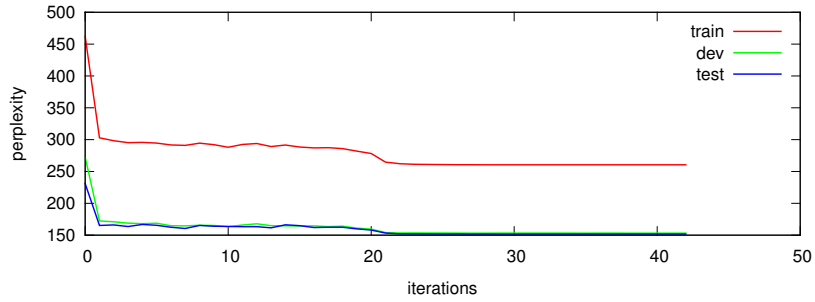


Figure 5.3: Perplexity curves over the iterations for monolingual clustering on the source side of IWSLT 2012 German→English train, dev, and test data.

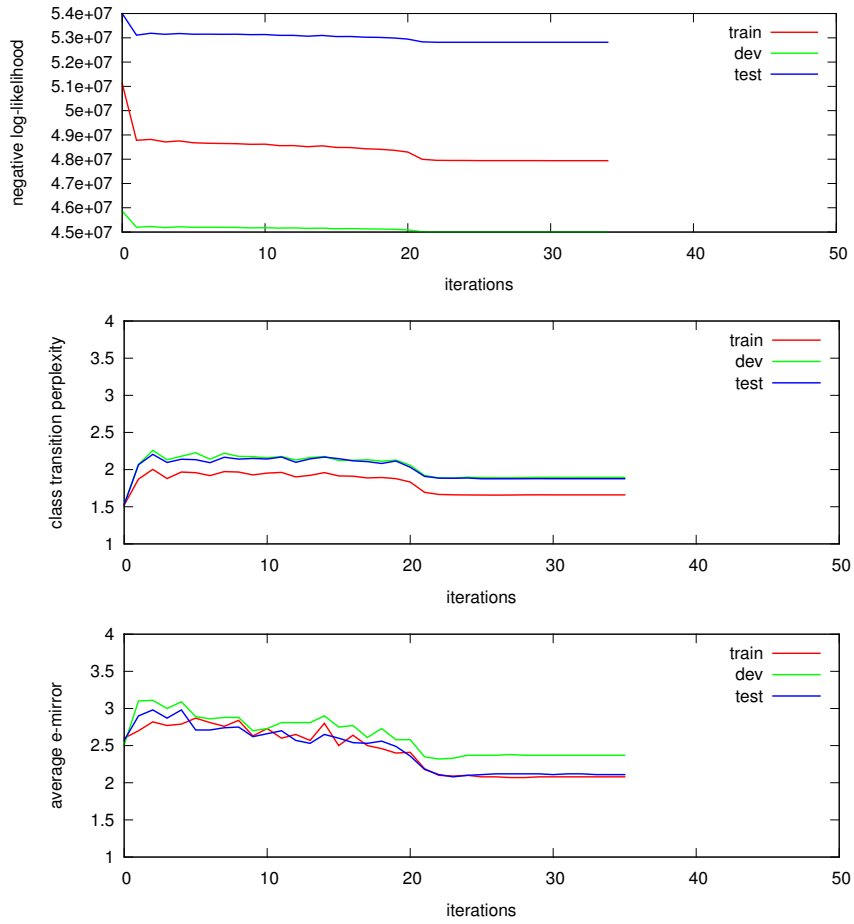


Figure 5.4: Clustering measure curves over the iterations for bilingual one-step clustering on IWSLT 2012 German→English train, dev, and test data: negative log-likelihood (top), class transition perlexity (middle), and average ϵ-mirror (bottom).

the same way for all three types of data: training, development, and test sets. There is no clear point in the curves that the measure value becomes lower only for the training data. Hence, the word clustering algorithms seem not to overfit to the data used for the optimization. This makes our parameter tuning process—only on the training data—convincing. The same curves for bilingual two-step clustering are shown in Figure A.2.

5.2.2 Translation Measures

Clustering evaluation in the previous section is *intrinsic*; the evaluation process is simple, but not explicitly related to the application of the clustering output. It is based on the belief that a good value in a clustering measure is carried over to a good performance in the translation usage, which is still vague.

For this reason, we also perform *extrinsic* evaluation of the word clusterings; the quality of a clustering is measured by its performance in translation. We build a complete translation system for each word class mapping, with which the word class models are trained. Accordingly, we check how much translation quality is improved by those word class models. We estimate various word class mappings and repeat this procedure to find the optimal mapping in terms of the effect on translation. BLEU is used for the translation metric in this procedure.

We carry out this optimization with regard to the following four factors:

1. **Clustering iterations.** In Section 5.2.1, it is shown that the number of iterations is the most influential factor in the clustering measures. Unlike other parameters, e.g. ν or γ, the measure values consistently change over the iterations. We now verify its effect on the translation quality of word class models.

   As we run the clustering algorithm, we extract the intermediate class mappings for each iteration and train wcTM and CSM source + target with it. For each iteration, the BLEU scores of the corresponding word class models are computed, which are given in Figure 5.3. First of all, when we tune the model weights separately for each iteration, the performance of the word class models is considerably fluctuating over the clustering iterations. We observe the maximum difference of 1.0 BLEU for the same model trained with different word class mappings.

   Next, we use a fixed set of model weights for all iterations—they are trained only once—in order to focus only on the change of word class mapping. The scores are nearly the same ($\pm$ 0.1%) over the iterations. Surprisingly, the word class models trained with the initial clustering, i.e. when the clustering algorithm does not even start yet, show the same level of performance with those trained with more optimized classes. This provides an important clue

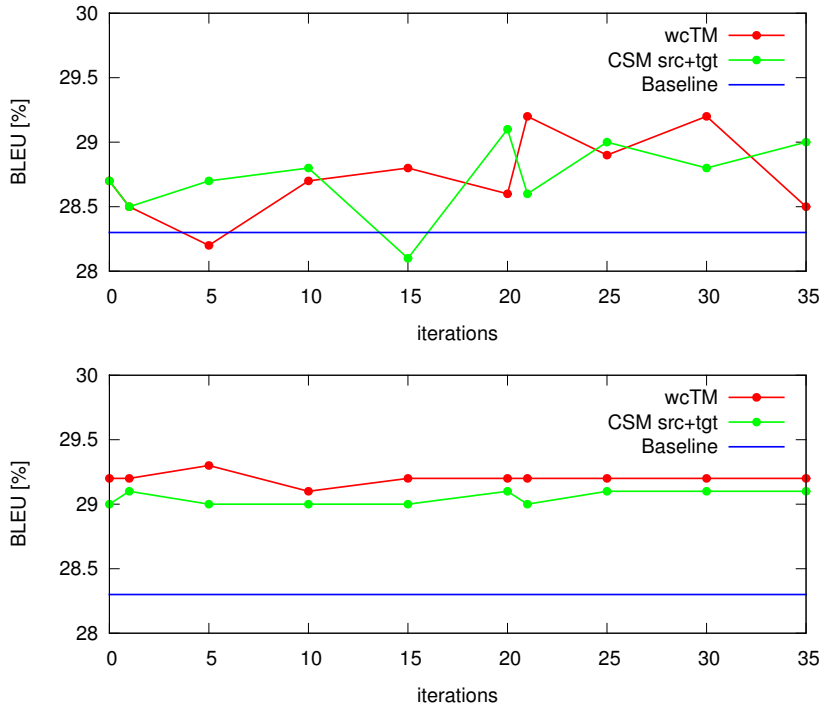


Figure 5.5: BLEU curves over the iterations for word class models on IWSLT 2012 German→English test data. The model weights are tuned separately for each iteration (top), or fixed for every iteration (bottom). Dots indicate those iterations which the translation is performed. 100 monolingual classes on both sides are used to train wcTM and CSM source + target.

that the whole process of the word clustering has no meaning at least when the output classes are used in phrase-based SMT—once we have an optimal set of model weights. The BLEU curves for bilingual clustering, which have the same tendency, are found in Section A.

2. **Initialization of the clustering.** If the clustering process has no significant impact on the translation quality, we can hypothesize that the initialization may dominate the clustering algorithm. We design five different initial clustering methods:

   - random: randomly assign words to classes
   - top-frequent: top frequent words have their own classes for each, all other words in the last class (used in the experiments of Figure 5.5)
   - same-countsum: each class has the same (or as similar as possible) sum of word counts

- same-#words: each class has the same (or as similar as possible) number of words
- count-bins: each class represents a bin of total count range

Table 5.4 shows the translation results with the word class models trained with these initializations—without running the clustering algorithm. The model weights are the same for every initialization. The translation scores are not changed much with different initial clusterings. We find that the initialization method does not affect the translation performance of the word class models. As an extreme case, random clustering is also a fine candidate for training word class models.

		dev		test	
System	Initialization	BLEU [%]	TER [%]	BLEU [%]	TER [%]
Baseline		30.3	50.0	28.3	52.2
+ wcTM	random	30.3	49.9	29.1	51.4
	top-frequent	30.8	48.9	29.2	50.8
	same-countsum	30.5	49.6	29.2	51.3
	same-#words	30.0	49.3	28.9	50.8
	count-bins	30.7	49.0	29.3	50.8
+ $\text{CSM}_{\text{src+tgt}}$	random	30.7	49.0	29.0	51.0
	top-frequent	30.8	48.7	29.0	50.8
	same-countsum	30.7	48.9	29.0	50.9
	same-#words	30.8	48.8	29.2	50.9
	count-bins	30.8	48.7	29.1	50.8

Table 5.4: Translation results for wcTM and CSM source + target with various initializations of the clustering. The experiments are done on IWSLT 2012 German→English data, and 100 monolingual classes are used on both sides.

3. **The number of classes.** It determines the granularity of the classes, which eventually adjusts the smoothing degree of the word class models. Table 5.5 gives the translation performance of wcTM and CSM source + target with varying number of classes. We run the clustering algorithm with default parameter values until the end, and the model weights are fixed for all settings. Similarly as before, increasing the number of classes is meaningless with respect to the translation quality of the word class models.

System	#classes	dev BLEU [%]	dev TER [%]	test BLEU [%]	test TER [%]
Baseline		30.3	50.0	28.3	52.2
+ wcTM	100	30.6	49.4	29.2	51.3
	200	31.0	49.1	29.0	51.3
	500	30.8	49.2	28.8	51.4
	1000	30.4	49.1	29.0	51.3
+ $\text{CSM}_{\text{src+tgt}}$	100	30.8	48.8	29.1	50.9
	200	30.8	49.1	28.9	51.6
	500	30.7	49.2	28.9	51.2
	1000	30.8	49.0	29.2	51.2

Table 5.5: Translation results for wcTM and CSM source + target with different numbers of classes. The experiments are done on IWSLT 2012 German→English data.

4. **Clustering algorithm.** The former three factors are optimized only with monolingual classes. Here, we estimate word classes with the bilingual algorithms and compare with the monolingual case (Table 5.6). Since we have learned that the number of classes does not influence on the translation results, we simply set it to 100 for efficiency. Contrary to our expectation, bilingual clustering algorithms have no advantage over monolingual clustering in the application to the word class models of phrase-based SMT.

System	Clustering Algorithm	dev BLEU [%]	dev TER [%]	test BLEU [%]	test TER [%]
Baseline		30.3	50.0	28.3	52.2
+ wcTM	monolingual	30.6	49.4	29.2	51.3
	bilingual-1step	31.1	48.6	29.3	50.9
	bilingual-2step	30.7	49.0	29.3	51.0
+ $\text{CSM}_{\text{src+tgt}}$	monolingual	30.8	48.8	29.1	50.9
	bilingual-1step	30.6	49.3	29.1	51.1
	bilingual-2step	30.9	49.0	29.0	51.0

Table 5.6: Translation results for wcTM and CSM source + target with different clustering algorithms. The experiments are done on IWSLT 2012 German→English data, and 100 monolingual classes are used on both sides.

In summary, the series of experiments show that the translation quality of the word class models does not significantly depend on the structure of word class mapping that we use to train them. No matter how we estimate the word class mapping, i.e. choosing the clustering iterations, initialization methods, the number of classes, and even the clustering algorithm itself, the word class models perform equivalently. The more important factor is the log-linear model training to find an optimal set of weights for the word class models. From this fact, we can save a substantial effort of optimizing the word class estimation when using the word class models in phrase-based SMT.

5.3 Word Class Models

In the previous section, we have already seen that wcTM and CSM source + target produce an improvement over the baseline system. Here, we systematically evaluate all word class models in Section 4.1. First off, we compare a diverse sort of CSM, which is our main contribution in the modeling problem, by their translation performance. Secondly, we measure the translation quality of wcTM, CSM, and wcLM on three different SMT tasks. We also validate the usefulness of their common refinement strategy: class membership probability. Finally, we combine different word class models to check if they create a synergy effect on the translation performance.

5.3.1 Comparison of CSM

CSM basically has four model types (CSM source, CSM source + target, CSM target, CSM target + source), depending on the sides and order where the class back-off is performed. They can be either linearly interpolated with the standard phrase translation model or log-linearly integrated into the existing model combination. In addition, CSMs can be categorized by its weighting schemes for each back-off position. For each factor in building a CSM, we investigate which choice is more effective in the translation quality.

Model Types and Integration

Table 5.7 shows the translation results for different types and integration methods of CSMs. CSM source + target performs the best overall on both dev and test among the four types. For the integration, the log-linear method yields better translation quality for the other three types, while the linear method is preferred for CSM source + target. We use linearly integrated CSM source + target—the single best setting of Table 5.7—throughout all experiments afterwards.

System	Integration	dev BLEU [%]	dev TER [%]	test BLEU [%]	test TER [%]
Baseline		30.3	50.0	28.3	52.2
+ CSM_{src}	linear	30.4	49.5	28.8	51.5
	log-linear	30.5	49.7	29.0	51.6
+ $\text{CSM}_{\text{src+tgt}}$	linear	**30.9**	**48.8**	**29.1**	**50.9**
	log-linear	30.8	48.6	28.9	50.6
+ CSM_{tgt}	linear	30.3	49.6	28.7	51.8
	log-linear	30.4	49.4	29.0	51.4
+ $\text{CSM}_{\text{tgt+src}}$	linear	29.9	50.1	28.2	52.1
	log-linear	30.7	48.9	29.1	50.9

Table 5.7: Translation results for various types of CSM on IWSLT 2012 German→English data. 100 monolingual classes are used on both sides.

Weighting Schemes

Table 5.8–5.10 gives the comparison between four CSM weighting schemes: equally weighted, inverse unigram, inverse source replacement probability, and factorization likelihood (Section 4.1.2). For IWSLT 2012 German→English data, no other schemes are better than equal weighting. In WMT 2014 English→German task, factorization likelihood shows the most promising result. We cannot clearly distinguish the performance difference between the weighting methods in WMT 2015 English→Czech data. As there is no single best scheme for all tasks, we can first stick to the simplest method (equal weighting) as default.

System	Weighting	dev BLEU [%]	dev TER [%]	test BLEU [%]	test TER [%]
Baseline		30.3	50.0	28.3	52.2
+ $\text{CSM}_{\text{src+tgt}}$	equal	**30.9**	**48.8**	**29.1**	**50.9**
	inverse-unigram	30.5	49.2	28.8	51.4
	inverse-replacement	30.8	48.8	29.1	50.9
	factorization	30.5	49.1	28.8	51.4

Table 5.8: Translation results for various weighting schemes of CSM on IWSLT 2012 German→English data. 100 monolingual classes are used on both sides.

System	Weighting	news-test2011 BLEU [%]	news-test2011 TER [%]	news-test2012 BLEU [%]	news-test2012 TER [%]	news-test2013 BLEU [%]	news-test2013 TER [%]
Baseline		12.6	73.8	13.4	71.8	14.6	69.8
+ $\text{CSM}_{\text{src+tgt}}$	equal	12.6	73.4	13.5	71.4	14.8	69.4
	inverse-unigram	12.7	73.4	13.6	71.1	14.9	69.3
	inverse-replacement	12.6	73.2	13.6	71.2	14.9	69.0
	factorization	**13.0**	**72.6**	**13.8**	**70.6**	**15.2**	**68.5**

Table 5.9: Translation results for various weighting schemes of CSM on WMT 2014 English→German data. 100 monolingual classes are used on both sides.

System	Weighting	news-test2012 BLEU [%]	news-test2012 TER [%]	news-test2013 BLEU [%]	news-test2013 TER [%]	news-test2014 BLEU [%]	news-test2014 TER [%]
Baseline		13.8	70.9	15.3	68.7	20.0	65.5
+ $\text{CSM}_{\text{src+tgt}}$	equal	13.7	70.9	15.4	68.6	20.4	65.1
	inverse-unigram	13.6	70.8	15.4	68.6	20.5	65.1
	inverse-replacement	13.7	70.7	15.5	68.2	20.5	64.9
	factorization	13.7	70.8	15.4	68.6	20.5	65.1

Table 5.10: Translation results for various weighting schemes of CSM on WMT 2015 English→Czech data. 100 monolingual classes are used on both sides.

5.3.2 Refinements with Class Membership Probability

In Section 4.1 we have presented the integration of class membership probability into wcTM, CSM, and wcLM. Table 5.11–5.13 shows the translation performance of each word class model and its class membership variant. In IWSLT 2012 German→English data, class membership probability slightly enhances wcLM and CSM source + target. The performance of wcTM is substantially improved (+0.4 BLEU and -1.6 TER on news-test2013) by this refinement. For WMT 2015 English→Czech data, we can see the improvement for all three word class models.

As supposed in Section 4.1.1, we claim that the improvements originate in that rare words are poorly scored due to their weak class membership. Consequently, the phrase pairs with the rare words are excluded from the translation candidates. This is particularly effective for the WMT datasets where numerous garbage words are included, e.g. special symbols, initials, or unusual compounds, from which many improbable phrase pairs are extracted. Indeed, class membership probability shows more distinct improvements in wcLM and wcTM on those datasets.

In all three datasets, the CSM achieves comparable performance to the wcTM. It should be noted that CSM requires a smaller number of features than wcTM:

System	dev BLEU [%]	dev TER [%]	test BLEU [%]	test TER [%]
Baseline	30.3	50.0	28.3	52.2
- wcLM + wcLM_{mbs}	30.2	49.2	**28.5**	**51.5**
+ wcTM	30.9	49.1	29.2	51.2
+ wcTM_{mbs}	30.7	49.2	29.2	51.2
+ $\text{CSM}_{\text{src+tgt}}$	30.9	48.8	29.1	50.9
+ $\text{CSM}_{\text{src+tgt:mbs}}$	30.8	48.8	**29.2**	**50.8**

Table 5.11: Translation results for class membership refinements on IWSLT 2012 German→English data. 100 monolingual classes are used on both sides.

System	news-test2011 BLEU [%]	news-test2011 TER [%]	news-test2012 BLEU [%]	news-test2012 TER [%]	news-test2013 BLEU [%]	news-test2013 TER [%]
Baseline	12.6	73.8	13.4	71.8	14.6	69.8
- wcLM + wcLM_{mbs}	12.3	74.0	13.3	72.0	14.6	69.7
+ wcTM	12.8	73.3	13.6	71.2	14.8	69.4
+ wcTM_{mbs}	**13.1**	**71.6**	13.5	69.9	**15.2**	**67.8**
+ $\text{CSM}_{\text{src+tgt}}$	12.6	73.4	13.5	71.4	14.8	69.4
+ $\text{CSM}_{\text{src+tgt:mbs}}$	12.7	73.3	13.5	71.3	14.8	69.2

Table 5.12: Translation results for class membership refinements on WMT 2014 English→German data. 100 monolingual classes are used on both sides.

System	news-test2012 BLEU [%]	news-test2012 TER [%]	news-test2013 BLEU [%]	news-test2013 TER [%]	news-test2014 BLEU [%]	news-test2014 TER [%]
Baseline	13.8	70.9	15.3	68.7	20.0	65.5
- wcLM + wcLM_{mbs}	13.4	70.2	**15.4**	**67.8**	**20.4**	**64.4**
+ wcTM	13.8	70.5	15.4	68.2	20.4	64.9
+ wcTM_{mbs}	13.7	69.9	**15.6**	**67.4**	**20.7**	**64.2**
+ $\text{CSM}_{\text{src+tgt}}$	13.7	70.9	15.4	68.6	20.4	65.1
+ $\text{CSM}_{\text{src+tgt:mbs}}$	13.7	70.8	15.4	68.5	**20.5**	**64.9**

Table 5.13: Translation results for class membership refinements on WMT 2015 English→Czech data. 100 monolingual classes are used on both sides.

two for CSM (the normal and the inverse direction) and four for wcTM (h_{wcPhr} and h_{wcLex}, the normal and the inverse direction for each). This is a crucial advantage for training the model weights with MERT, since the optimization becomes unstable for large amount of models. Compared to wcTM, CSM offers extra room for other additional models to be combined without degrading the MERT procedure.

Notes on wcLM and Class Membership Probability

With the help of class membership probability, wcLM can be factorized over words instead of word classes (Equation 4.15). This makes it possible to compare its perplexity with that of a conventional language model defined on a word vocabulary. Table 5.14 gives the perplexity values for three kinds of target language models: the normal word language model (LM), wcLM with class membership probability ($\text{wcLM}_{\mathsf{mbs}}$), and the linear interpolation between them. It is hardly surprising that $\text{wcLM}_{\mathsf{mbs}}$ has worse perplexity than LM, since it is trained on a coarser vocabulary. However, when it is combined with LM, the perplexity eventually decreases. This provides another solid ground for the usage of wcLM along with LM.

Dataset	Perplexity		
	LM	$\text{wcLM}_{\mathsf{mbs}}$	LM + $\text{wcLM}_{\mathsf{mbs}}$
IWSLT 2012 German→English	105.43	263.27	104.08
WMT 2014 English→German	636.67	1206.71	510.83
WMT 2015 English→Czech	624.88	1851.37	500.72

Table 5.14: Perplexity values for the target word language model and the wcLM (equipped with class membership probability) on different datasets. 100 monolingual classes are used.

We perform an additional set of experiments for wcLM with varying number of classes. The corresponding change of the perplexity and the translation performance is shown in Table 5.15.

#classes	Perplexity	BLEU [%]
100	263.27	28.5
200	231.70	28.4
500	193.58	29.1
1000	178.49	29.2

Table 5.15: Perplexity and translation results for $\text{wcLM}_{\mathsf{mbs}}$ with varying number of classes. BLEU scores are measured on IWSLT 2012 German→English test, when the corresponding wcLM replaces that of the baseline.

As expected, the perplexity value gets lower by increasing the number of classes, which means we use more fine-grained vocabulary to train the wcLM. A notable result is that its performance in translation also improves over the number of classes. This is utterly opposed to the result of Section 5.2.2, where the number of classes does not affect the translation quality of wcTM or CSM.

5.3.3 Model Combinations

Since each word class model individually accomplishes a decent translation quality, we consider the combination of different word class models, expecting to boost the performance. Tables 5.16 shows the translation results of combined word class models on IWSLT 2012 German→English task. When wcTM and CSM are used together, we observe a substantial improvement on dev—where the model weights are optimized—but it does not carry over to test. It is difficult to conclude that the combination of different translation models is generally helpful.

On the other hand, wcHRM clearly generates a synergy with wcTM, producing the best result on this dataset so far. wcHRM delivers reordering information which is completely different from wcTM or CSM, thus the combination with the translation models enriches the system.

System	dev BLEU [%]	dev TER [%]	test BLEU [%]	test TER [%]
Baseline	30.3	50.0	28.3	52.2
+ wcTM_{mbs}	30.7	49.2	29.2	51.2
+ $\text{CSM}_{\text{src:mbs}}$	31.1	48.7	29.2	51.2
+ $\text{CSM}_{\text{src+tgt:mbs}}$	30.6	49.1	29.1	51.2
+ $\text{CSM}_{\text{src:mbs}}$ + $\text{CSM}_{\text{src+tgt:mbs}}$	31.2	48.6	29.2	50.8
+ wcHRM	**31.5**	**48.1**	**29.5**	**50.6**

Table 5.16: Translation results for the combination of word class models on IWSLT 2012 German→English data. The indentation level in System column indicates that the model is added on top of the above setting. 100 monolingual classes are used on both sides.

5.3.4 Translation Examples

We provide a couple of translation outputs, which are actually enhanced by the word class models, in Table 5.17.

Source	unsere Zeit Schriften werden von Millionen **gelesen** .
Reference	our magazines are **read** by millions .
Baseline	our magazines are **killed** by millions .
+ $CSM_{src+tgt}$	our magazines will **read** from millions .

Source	viele von ihnen sind von Gesichtern **verdeckt** usw.
Reference	so many of them are **occluded** by faces , and so on .
Baseline	many of them are of faces , and so on .
+ $CSM_{src+tgt}$	many of them are **covered** by faces , and so on .

Source	ich bekomme im wahren Leben dieses Feed back **nicht** .
Reference	I **don 't** get that feedback in real life .
Baseline	I **'m going to** get in the real life of this feedback .
+ $CSM_{src+tgt}$	I **don 't** get that feedback in real life .

Table 5.17: Translation examples of IWSLT 2012 German→English data, where CSM source + target improves the quality.

The first example shows how an unlikely translation is demoted by CSM. The baseline translation is nonsense due to the word "killed", which has nothing to do with the source word "gelesen". The translation rule between them is learned from an inaccurate word alignment. If we back off to the word classes, "gelesen" is not to be translated to the class of "killed", i.e. any related word of "killed". By giving a low score to "killed", CSM selects the correct target translation "read".

The second example shows how a low-score translation rule is promoted by the word class smoothing. The baseline system fails to find a proper translation for "verdeckt". CSM cannot translate it to the word "occluded" as the reference, but to a semantically similar word "covered". The two target words are in the same class, so the word class smoothing gives a high score to the translation from "verdeckt" to the class of "covered".

In the last example, CSM behaves in a similar way as in the first example. It devalues the phrase "'m going to" and encourages "don 't" for the translation of "nicht". As a result, its translation output is the same as the reference sentence.

5.4 Word Class Decoding

The paraphrasing method for phrase-based SMT decoding is proposed in Section 4.2. We can borrow the class back-off phrase pairs from CSMs and use them as

System	τ_u	N_h	dev BLEU [%]	dev TER [%]	dev #CSM	test BLEU [%]	test TER [%]	test #CSM
Baseline			30.3	50.0		28.3	52.2	
wcDec	max	500	29.2	50.5		27.9	52.1	
wcDec (w/o OOV)	max	500	**30.6**	**49.2**	49	**29.2**	**51.0**	117
	1000	200	30.4	49.4	0	28.7	51.7	1

Table 5.18: Translation results for the word class decoding (wcDec) on IWSLT 2012 German→English data. CSM source model is used for the paraphrasing. 2000 monolingual classes are used for wcDec, and 100 monolingual classes for wcDec w/o OOV. The number of CSM phrase pairs actually used in the translation (#CSM) is also reported.

alternative translation options. Here, we show only preliminary experiments using CSM source on IWSLT 2012 (Table 5.18) due to the time limit.

To handle OOV with word classes, we estimate the word class mapping from a huge collection of German monolingual data, provided by WMT 2014 translation task. Setting parameter τ_u to the maximum possible value, we let every source word considered in the paraphrasing procedure. Unfortunately, the word class decoding shows an inferior performance than the baseline decoding. We attribute this result to the excessively large vocabulary of the monolingual data used for the clustering. A large vocabulary leads to too many paraphrases, which might make the search space too complicated. Although we set the number of classes relatively large (= 2000), each class should still cover lots of words that are not closely related to each other. This fact can degrade the overall quality of the paraphrases.

As a next step, we exclude OOV words from the translatable input in the word class decoding. In this case, we may learn the word classes only from the given bilingual data, which produces more compact and meaningful classes. We obtain a fine improvement (+0.9 BLEU and -1.2 TER on test) over the baseline by performing the paraphrasing except for OOV words. We decrease τ_u and N_h to reduce the search space, but the translation quality becomes worse and almost no CSM phrase pairs are actually used.

Translation Examples

Table 5.19 shows some translation examples of the word class decoding. In the first example, the baseline system produces a weird translation, where an unnecessary passive voice confuses the sentence structure. Using the word class decoding, we obtain a grammatically correct translation which is very similar to the reference.

Source	wir haben die Biologie digitalisiert ...
Reference	we **'ve been digitizing** biology ...
Baseline	we 've got the biology **is being digitized**
wcDec	we **have digitized** biology

Source	... mit der wir diese kleinen Stücke zusammensetzen und die Fehler korrigieren konnten .
Reference	... for putting these little pieces together and **correct all the errors .**
Baseline	... with which we have these little pieces , and **the correct mistakes .**
wcDec	... with which we have these little pieces together and **to correct the mistakes .**

Table 5.19: Translation examples of IWSLT 2012 German→English data, where the word class decoding improves the quality. Underlined is where a CSM phrase pair is used.

CSM phrase pair `$C74 Biologie` → `biology` is used here, which is not directly related to the improved part in the sentence. We suppose that a CSM phrase pair adjust the search space in such a way that the decoder more easily finds the correct translation and reordering of the neighboring phrase.

We can observe a similar effect in the second example. The baseline translation distorts the intended meaning by putting an article "`the`" in the wrong place. CSM phrase pair `$C74 Fehler` → `the mistakes` locates the article in the right position, making the translation closer in meaning to the reference.

In the given examples, the parts translated with CSM phrase pairs can also be translated with original phrase pairs; the rules `die Biologie` → `biology` (for the first example) and `die Fehler` → `the mistakes` (for the second example) already exist in the original phrase table. Considering that a frequent word (an article) is backed off to its class, this is predictable. Contrary to our expectation, the word class decoding here improves the translation quality not by providing diversity in translation options, but by rearranging the search space of the existing translation options.

5.5 Word Class Alignments

We have presented the training and usages of word class alignments in Section 4.3. Here, we learn the word class alignments on IWSLT 2012 German→English data and evaluate their effect on the translation quality.

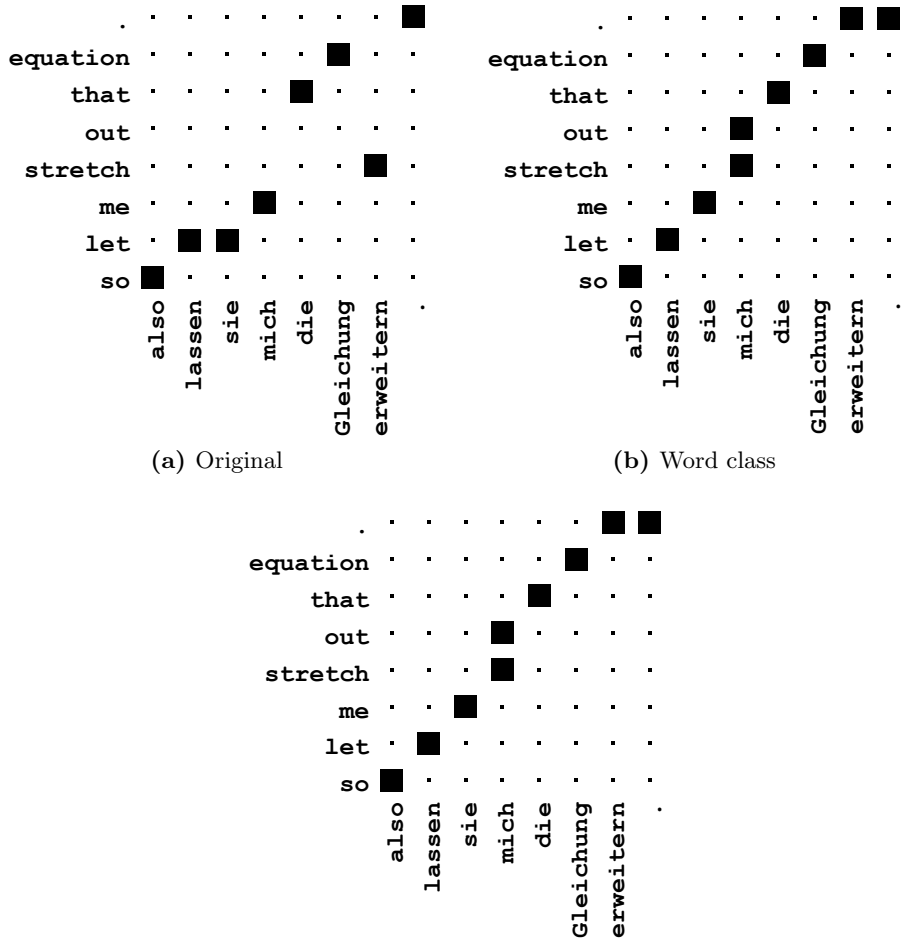
System	Alignment	dev BLEU [%]	dev TER [%]	test BLEU [%]	test TER [%]
Baseline	original	30.3	50.0	28.3	52.2
	merged	29.6	50.4	27.7	52.8
+ wcTM	original	30.9	49.1	29.2	51.2
	merged	29.9	50.1	27.7	52.4
+ CSM_{src}	original	30.5	49.7	29.0	51.6
	merged	29.7	49.9	27.9	52.4
+ $\text{CSM}_{\text{src+tgt}}$	original	30.8	48.9	28.8	51.0
	merged	29.2	50.8	27.3	53.1

Table 5.20: Translation results with the merged alignment between the original and word class alignments on IWSLT 2012 German→English data. 100 monolingual classes are used on both sides.

System	Alignment	dev BLEU [%]	dev TER [%]	test BLEU [%]	test TER [%]
Baseline	original	30.3	50.0	28.3	52.2
	merged	30.2	49.8	28.4	51.9
+ wcTM	original	30.6	49.4	29.2	51.3
	merged	29.9	50.1	28.2	51.9
+ CSM_{src}	original	30.5	49.7	29.0	51.6
	merged	29.5	50.2	28.1	52.1
+ $\text{CSM}_{\text{src+tgt}}$	original	30.8	48.8	29.1	50.9
	merged	30.1	50.2	28.2	52.2

Table 5.21: Translation results with the merged alignment between the original and word class alignments on IWSLT 2012 German→English data. 1000 monolingual classes are used on both sides.

In Table 5.20, we first present the performance change in the baseline models and the word class models, when the word class alignment is merged with the original alignment. The results show that the merged alignment causes a clear degradation of the translation quality in all settings. Figure 5.6 shows an example of incorrect word class alignments and their effect on the merged alignment.



(a) Original **(b)** Word class

(c) Merged

Figure 5.6: A German→English bilingual sentence pair with different word alignments: (a) the original alignment, (b) word class alignment, and (c) the merged alignment between the two. German words "sie", "mich", and "erweitern" are aligned to the irrelevant English words in (b) and (c). Taken from IWSLT 2012 German→English data.

We argue two reasons for this performance loss. Firstly, too many words are assigned to each class, making the classes and their translation counts linguistically meaningless. Secondly, the merging heuristic is not intelligent enough to capture only the beneficial information of both alignments. From the first point, we increase the number of classes on both sides from 100 to 1000 and run the same set of experiments (Table 5.21). The results are slightly better (+0.2–0.7 BLEU) than

the case of 100 classes but still far worse than using the original alignment. We can suppose that much larger number of classes, e.g. 10000 or 20000, are necessary to make this approach effective. In such cases, however, the word class estimation and the word alignment training become extremely inefficient, which is not attractive. Addressing the second reason is out of the scope of this work.

We now turn to another method of using word class alignments: training word class models with them. As explained in 4.3, we need penalty costs for the word class model scores computed with the original phrase counts. In this experiment, we manually set a fixed penalty to all such cases according to the score statistics of the word class models. Figure 5.7 shows an exemplary distribution of a word class model for our experiments. The scores are in negative log scale, so the lower means better. The distribution follows the similar structure, regardless of the alignment we use to learn the model. We conclude that +5 is a reasonable value for the penalty; more than half of the scores from the original alignment are eventually ignored in the translation, but some very decent scores of them remain still competitive with the scores from the word class alignment.

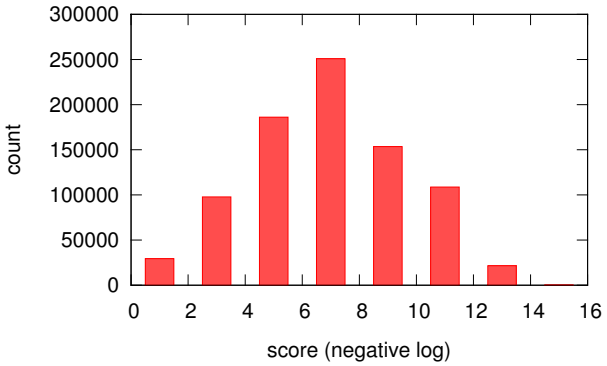


Figure 5.7: Histogram of wcTM source-to-target scores on IWSLT 2012 German→English data, trained with the original word alignment. 100 monolingual classes are used on both sides.

Table 5.22 and 5.23 show the results with 100 classes and 1000 classes, respectively. For both cases, word class alignments enhance the performance of the word class models. More fine-grained classes are beneficial for wcTM and CSM source, but still do not make the models better than those from the original training.

System	Alignment	dev BLEU [%]	dev TER [%]	test BLEU [%]	test TER [%]
Baseline	original	30.3	50.0	28.3	52.2
+ wcTM	original	30.6	49.4	29.2	51.3
	word class	30.8	49.2	28.7	51.5
+ CSM_{src}	original	30.5	49.7	29.0	51.6
	word class	30.1	49.9	28.3	51.9
+ $\text{CSM}_{\text{src+tgt}}$	original	30.8	48.8	29.1	50.9
	word class	30.4	49.5	28.4	51.7

Table 5.22: Translation results for the word class models trained with the word class alignment on IWSLT 2012 German→English data. 100 monolingual classes are used on both sides.

System	Alignment	dev BLEU [%]	dev TER [%]	test BLEU [%]	test TER [%]
Baseline	original	30.3	50.0	28.3	52.2
+ wcTM	original	30.6	49.4	29.2	51.3
	word class	30.9	49.1	29.0	51.4
+ CSM_{src}	original	30.5	49.7	29.0	51.6
	word class	30.7	49.3	29.0	51.4
+ $\text{CSM}_{\text{src+tgt}}$	original	30.8	48.8	29.1	50.9
	word class	30.0	50.2	28.4	52.1

Table 5.23: Translation results for the word class models trained with the word class alignment on IWSLT 2012 German→English data. 1000 monolingual classes are used on both sides.

Chapter 6

Conclusion

6.1 Summary

In this thesis work, we have investigated the usages of word classes in modeling, decoding, and word alignment problems of phrase-based SMT.

For the modeling, we have developed a novel smoothing method for the standard phrase translation model called CSM. It performs the back-off from each word in a phrase to its corresponding word class, which is done one word at a time and take the average with various weighting schemes. On three different translation tasks (IWSLT 2012 German→English, WMT 2014 English→German, and WMT 2015 English→Czech), CSM improves the translation quality over the strong baseline by up to +0.9 BLEU and -1.4 TER. This is at least the equal level of performance to the state-of-the-art word class model (wcTM) with a smaller number of features.

In addition, we have validated the utility of class membership probability in the word class models. By reformulating with the class membership probability, the translation performance of CSM, wcTM, and wcLM is enhanced by up to +0.4 BLEU and -1.6 TER in our experiments.

Furthermore, we have extensively compared various word class mappings with respect to intrinsic clustering measures and extrinsic performance in the word class models. Our results show that the performance of wcTM and CSM is not significantly affected by the word class mapping they are trained with, regardless of clustering parameters, initial clustering, and even the clustering algorithm itself for its estimation. We show that the performance is greatly dependent on the log-linear training of the model weights.

On the other hand, we have verified that wcLM can be further improved by increasing the number of classes. On IWSLT 2012 German→English data, the translation quality of wcLM is enhanced by +0.7 BLEU when the number of classes are changed from 100 to 1000.

When different word class models are combined, wcTM and CSM do not show a synergy effect on the translation performance. We show that wcHRM is effective to be used together with the translation models.

For the decoding, we have proposed a class-based paraphrasing method to provide additional translation options, using the phrase pairs from CSM. We have integrated this procedure to the standard phrase-based SMT decoder. Our modified decoder shows an improvement by up to +0.9 BLEU and -1.2 TER on IWSLT 2012 German→English task.

For the word alignment problem, we have presented a simple method to train word alignments from a word class corpus. We have applied the word class alignment to train the standard phrase-based SMT models and also the word class models. Unfortunately, the word class alignments do not show a superior performance to the original word alignments in all of our experiments.

6.2 Future Work

CSM and wcTM are both designed within the phrase level, which cannot capture the context beyond the phrase boundaries. There may be a limit in their performance gain by only the refinements and combinations. As a next step, we may develop a word class model which provides useful information about longer contexts. For this purpose, we can use word class sequences as inputs to neural networks, which are successfully applied to SMT recently for modeling sentence-level contexts.

We have found that the performance of wcLM largely depends on the word class estimation. It would be meaningful to compare various word classes for the usage in wcLM, similar to Section 5.2. Furthermore, we may interpolate many different wcLMs each of which is trained with different word class mappings.

Regarding the word class decoding, a number of empirical evaluations are still needed. The paraphrasing with CSM target direction models is to be tested, and there should be a further study on the parameters τ_u and N_h. Since the modified decoder does not help with OOV translations in our experiments, we need to find an effective refinement for OOV handling. For example, we may find another method to learn a useful class mapping for OOV words—not just employing a massive monolingual dataset.

Appendix A

Optimizing Bilingual Classes

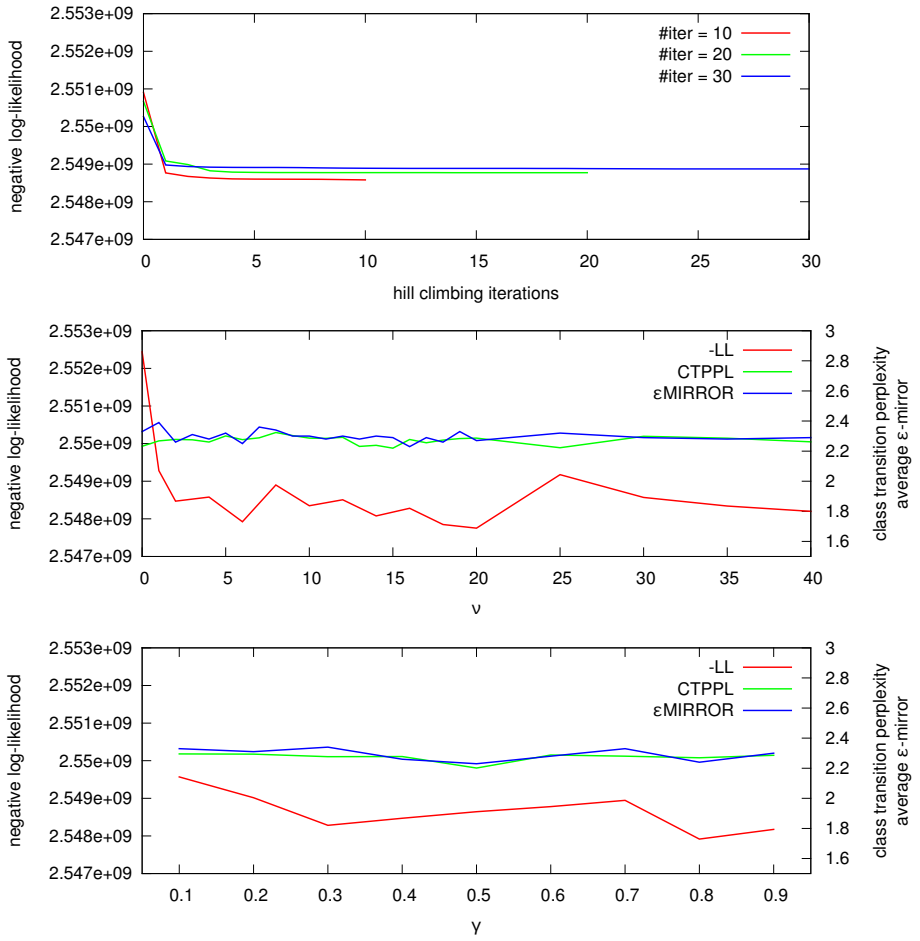


Figure A.1: Clustering measure curves over hill climbing phase iterations (top), ν (middle), and γ (bottom) for bilingual two-step clustering on WMT 2014 English→German data. The top figure has negative log-likelihood curves for different values of #iter. The middle and bottom figures have individual curves for different clustering measures.

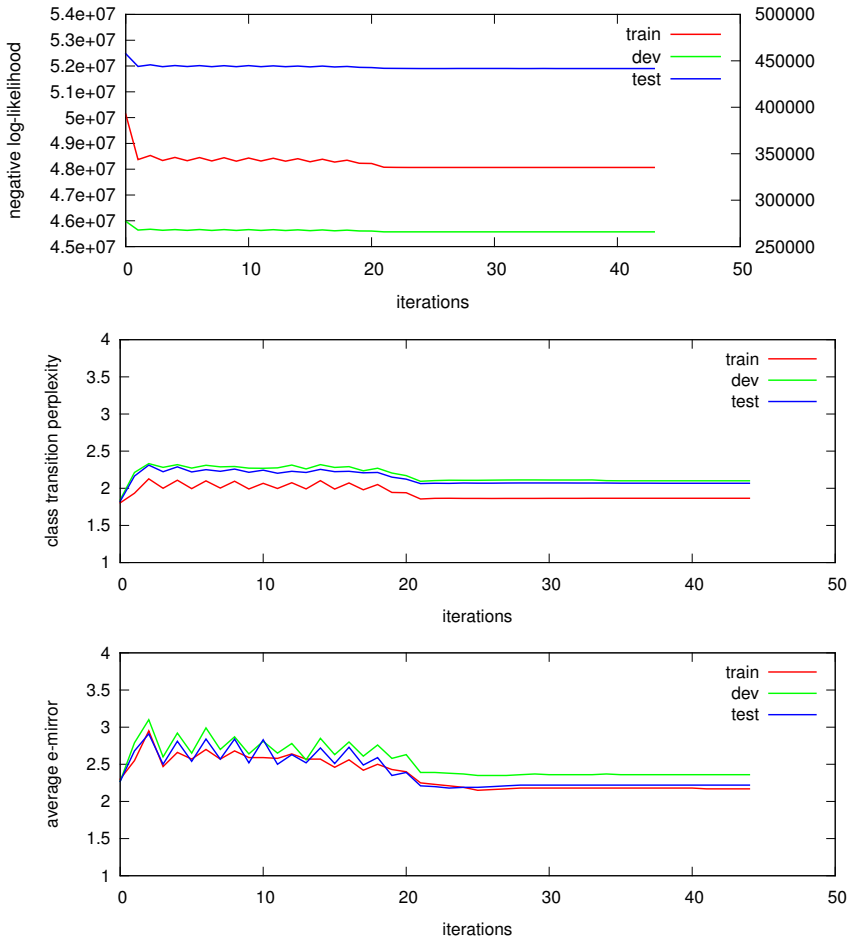


Figure A.2: Clustering measure curves over the iterations for bilingual two-step clustering on IWSLT 2012 German→English train, dev, and test data: negative log-likelihood (top), class transition perlexity (middle), and average ϵ-mirror (bottom).

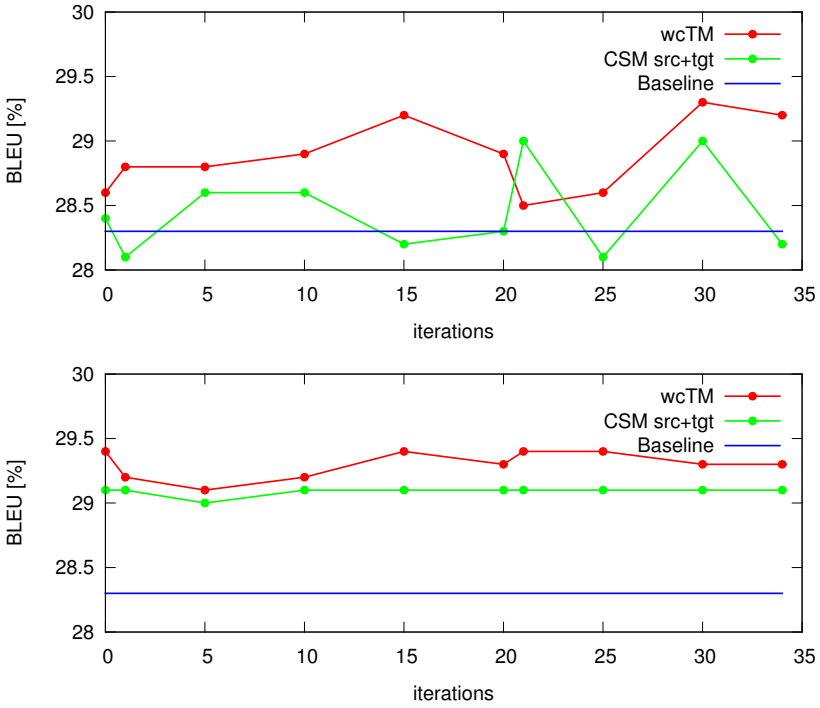


Figure A.3: BLEU curves over the iterations for word class models on IWSLT 2012 German→English test data. The model weights are tuned separately for each iteration (top), or fixed for every iteration (bottom). Dots indicate those iterations which the translation is performed. 100 bilingual one-step classes on both sides are used to train wcTM and CSM source + target.

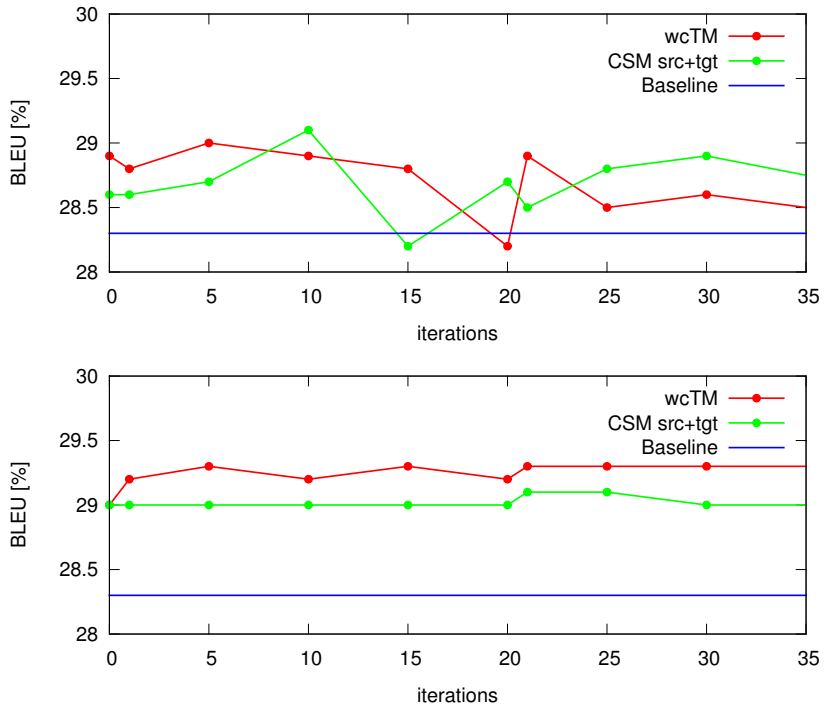


Figure A.4: BLEU curves over the iterations for word class models on IWSLT 2012 German→English test data. The model weights are tuned separately for each iteration (top), or fixed for every iteration (bottom). Dots indicate those iterations which the translation is performed. 100 bilingual two-step classes on both sides are used to train wcTM and CSM source + target.

List of Figures

List of Tables

Bibliography

[Bellman 57] R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, USA, 1957.

[Berger & Pietra$^+$ 96] A. L. Berger, S. A. D. Pietra, V. J. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, Vol. 22, No. 1, pp. 39–71, March 1996.

[Bisazza & Monz 14] A. Bisazza, C. Monz. Class-based language modeling for translating into morphologically rich languages. In *25th International Conference on Computational Linguistics (COLING 2014)*, pp. 1918–1927, Dublin, Ireland, August 2014.

[Borthwick & Sterling$^+$ 98] A. Borthwick, J. Sterling, E. Agichtein, R. Grishman. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *6th Workshop on Very Large Corpora (VLC 1998)*, pp. 152–160, Montreal, Canada, August 1998.

[Botros 15] R. Botros. Efficient training of word classes and its application to language modeling. Master's thesis, Department of Computer Science, RWTH Aachen University, Aachen, Germany, February 2015.

[Brody & Laputa 09] S. Brody, M. Laputa. Bayesian word sense induction. In *12th Conference of the European Chapter of the ACL (EACL 2009)*, pp. 103–111, Athens, Greece, April 2009.

[Brown & deSouza$^+$ 92] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, J. C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, Vol. 18, No. 4, pp. 467–479, December 1992.

[Brown & Pietra$^+$ 93] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, Vol. 19, No. 2, pp. 263–311, June 1993.

[Cettolo & Girardi$^+$ 12] M. Cettolo, C. Girardi, M. Federico. Wit^3: Web inventory of transcribed and translated talks. In *16th Annual Conference of the European Association for Machine Translation (EAMT 2012)*, pp. 261–268, Trento, Italy, May 2012.

[Chahuneau & Schlinger$^+$ 13] V. Chahuneau, E. Schlinger, C. Dyer, N. A. Smith. Translating into morphologically rich languages with synthetic phrases. In *2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pp. 1677–1687, Seattle, WA, USA, October 2013.

[Cherry 13] C. Cherry. Improved reordering for phrase-based translation using sparse features. In *2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pp. 22–31, Atlanta, GA, USA, June 2013.

[Chiang & Knight$^+$ 09] D. Chiang, K. Knight, W. Wang. 11,001 new features for statistical machine translation. In *2009 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2009)*, pp. 218–226, Boulder, CO, USA, May 2009.

[Chiang 07] D. Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, Vol. 33, No. 2, pp. 201–228, June 2007.

[Chrupała 11] G. Chrupała. Efficient induction of probabilistic word classes with lda. In *5th International Joint Conference on Natural Language Processing (IJCNLP 2011)*, pp. 363–372, Chiang Mai, Thailand, November 2011.

[Collins & Koehn$^+$ 05] M. Collins, P. Koehn, I. Kučerová. Clause restructuring for statistical machine translation. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pp. 531–540, Ann Arbor, MI, USA, June 2005.

[Deerwester & Dumais$^+$ 90] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, Vol. 41, No. 6, pp. 391–407, 1990.

[Dhillon & Foster$^+$ 11] P. S. Dhillon, D. Foster, L. Ungar. Multi-view learning of word embeddings via cca. In *25th Annual Conference on Neural Information Processing Systems (NIPS 2011)*, pp. 199–207, Granada, Spain, December 2011.

[Dinu & Laputa 10] G. Dinu, M. Laputa. Measuring distributional similarity in context. In *2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1162–1172, Cambridge, MA, USA, October 2010.

[Emami & Jelinek 05] A. Emami, F. Jelinek. Random clusterings for language modeling. In *30th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2005)*, pp. 581–584, Philadelphia, PA, USA, March 2005.

[Faruqui & Dyer 13] M. Faruqui, C. Dyer. An information theoretic approach to bilingual word clustering. In *51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pp. 777–783, Sofia, Bulgaria, August 2013.

[Foster & Kuhn$^+$ 06] G. Foster, R. Kuhn, H. Johnson. Phrasetable smoothing for statistical machine translation. In *2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pp. 53–61, Sydney, Austrailia, July 2006.

[Fraser & Weller$^+$ 12] A. Fraser, M. Weller, A. Cahill, F. Cap. Modeling inflection and word-formation in smt. In *13th Conference on European Chapter of the Association for Computational Linguistics (EACL 2012)*, pp. 664—-674, Avignon, France, April 2012.

[Galley & Manning 08] M. Galley, C. D. Manning. A simple and effective hierarchical phrase reordering model. In *2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pp. 848–856, Honolulu, HI, USA, October 2008.

[Galley & Manning 10] M. Galley, C. D. Manning. Accurate non-hierarchical phrase-based translation. In *2010 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT 2010)*, pp. 966–974, Los Angeles, CA, USA, June 2010.

[Goldwater & Griffiths 07] S. Goldwater, T. L. Griffiths. A fully bayesian approach to unsupervised part-of-speech tagging. In *45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, pp. 744–751, Prague, Czech Republic, June 2007.

[Green & DeNero 12] S. Green, J. DeNero. A class-based agreement model for generating accurately inflected translations. In *50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pp. 146–155, Jeju, Republic of Korea, July 2012.

[Green & Wang$^+$ 13] S. Green, S. Wang, D. Cer, C. D. Manning. Fast and adaptive online training of feature-rich translation models. In *51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pp. 311–321, Sofia, Bulgaria, August 2013.

[Harris 54] Z. S. Harris. Distributional structure. *Word*, Vol. 10, No. 23, pp. 146–162, 1954.

[Hopkins & May 11] M. Hopkins, J. May. Tuning as ranking. In *2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pp. 1352–1362, Edinburgh, Scotland, July 2011.

[Huang & Socher+ 12] E. H. Huang, R. Socher, C. D. Manning, A. Y. Ng. Improving word representations via global context and multiple word prototypes. In *50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pp. 873–882, Jeju, Republic of Korea, July 2012.

[Jelinek 98] F. Jelinek. *Statistical Methods for Speech Recognition.* MIT Press, Cambridge, MA, USA, 1998.

[Khudanpur & Wu 00] S. Khudanpur, J. Wu. Maximum entropy techniques for exploiting syntactic, semantic, and collocational dependencies in language modeling. *Computer Speech and Language*, Vol. 14, No. 4, pp. 355–372, October 2000.

[Kneser & Ney 93] R. Kneser, H. Ney. Improved clustering techniques for class-based statistical language modelling. In *3rd European Conference on Speech Communication and Technology (EUROSPEECH 1993)*, pp. 973–976, Berlin, Germany, September 1993.

[Knight 99] K. Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, Vol. 25, No. 4, pp. 607–615, December 1999.

[Koehn & Hoang 07] P. Koehn, H. Hoang. Factored translation models. In *2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pp. 868–876, Prague, Czech Republic, June 2007.

[Koehn & Och+ 03] P. Koehn, F. J. Och, D. Marcu. Statistical phrase-based translation. In *2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT 2003)*, pp. 48–54, Edmonton, Canada, May 2003.

[Koehn 05] P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *10th Machine Translation Summit (MT Summit X)*, pp. 79–86, Phuket, Thailand, September 2005.

[Koo & Carreras+ 08] T. Koo, X. Carreras, M. Collins. Simple semi-supervised dependency parsing. In *46th Annual Meeting of the Association for Computational Linguistics (ACL 2008)*, pp. 595–603, Columbus, OH, USA, June 2008.

[Liang 05] P. Liang. Semi-supervised learning for natural language. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, May 2005.

[Marcus & Marcinkiewicz$^+$ 93] M. P. Marcus, M. A. Marcinkiewicz, B. Santorini. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, Vol. 19, No. 2, pp. 313–330, June 1993.

[Martin & Liermann$^+$ 98] S. Martin, J. Liermann, H. Ney. Algorithms for bigram and trigram word clustering. *Speech Communication*, Vol. 24, No. 1, pp. 19–37, 1998.

[Merialdo 94] B. Merialdo. Tagging english text with a probabilistic model. *Computational Linguistics*, Vol. 20, No. 2, pp. 155–171, June 1994.

[Mikolov & Sutskever$^+$ 13] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean. Distributed representations of words and phrases and their compositionality. In *27th Annual Conference on Neural Information Processing System (NIPS 2013)*, pp. 3111–3119, Lake Tahoe, NV, USA, December 2013.

[Miller & Guinness$^+$ 04] S. Miller, J. Guinness, A. Zamanian. Name tagging with word clusters and discriminative training. In *2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2004)*, pp. 337–342, Boston, MA, USA, May 2004.

[Och & Ney 00] F. J. Och, H. Ney. Improved statistical alignment models. In *38th Annual Meeting of the Association for Computational Linguistics (ACL 2000)*, pp. 440–447, Hongkong, China, October 2000.

[Och & Ney 02] F. J. Och, H. Ney. Discriminative training and maximum entropy models for statistical machine translation. In *40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pp. 295–302, Philadelphia, PA, USA, July 2002.

[Och & Ney 03] F. J. Och, H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, Vol. 29, No. 1, pp. 19–51, March 2003.

[Och & Ney 04] F. J. Och, H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, Vol. 30, No. 4, pp. 417–449, December 2004.

[Och 95] F. J. Och. Maximum-likelihood-schätzung von wortkategorien mit verfahren der kombinatorischen optimierung. Studienarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany, May 1995.

[Och 99] F. J. Och. An efficient method for determining bilingual word classes. In *9th Conference on European Chapter of Association for Computational Linguistics (EACL 1999)*, pp. 71–76, Bergen, Norway, June 1999.

[Och 02] F. J. Och. *Statistical Machine Translation: From Single-Word Models To Alignment Templates*. Ph.D. thesis, Department of Computer Science, RWTH Aachen University, Aachen, Germany, October 2002.

[Och 03] F. J. Och. Minimum error rate training in statistical machine translation. In *41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pp. 160–167, Sapporo, Japan, July 2003.

[Papineni & Roukos$^+$ 02] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pp. 311–318, Philadelphia, PA, USA, July 2002.

[Popovič & Ney 06] M. Popovič, H. Ney. Pos-based word reorderings for statistical machine translation. In *5th International Conference on Language Resources and Evaluation (LREC 2006)*, pp. 1278–1283, Genoa, Italy, May 2006.

[Ratnaparkhi 96] A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *1996 Conference on Empirical Methods in Natural Language Processing (EMNLP 1996)*, Philadelphia, PA, USA, May 1996.

[Schütze 98] H. Schütze. Automatic word sense discrimination. *Computational Linguistics*, Vol. 24, No. 1, pp. 97–123, March 1998.

[Snover & Dorr$^+$ 06] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul. A study of translation edit rate with targeted human annotation. In *7th Conference of the Association for Machine Translation in the Americas (AMTA 2006)*, pp. 223–231, Cambridge, MA, USA, August 2006.

[Steinbiss & Tran$^+$ 94] V. Steinbiss, B.-H. Tran, H. Ney. Improvements in beam search. In *International Conference on Spoken Language Processing (ICSLP 1994)*, pp. 2143–2146, Yokohama, Japan, September 1994.

[Stolcke 02] A. Stolcke. Srilm – an extensible language modeling toolkit. In *International Conference on Spoken Language Processing (INTERSPEECH 2002)*, pp. 901–904, Denver, CO, USA, September 2002.

[Täckström & McDonald$^+$ 12] O. Täckström, R. McDonald, J. Uszkoreit. Cross-lingual word clusters for direct transfer of linguistic structure. In *2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2012)*, pp. 11–21, Montréal, Canada, June 2012.

[Tillmann 04] C. Tillmann. A unigram orientation model for statistical machine translation. In *2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Short Paper (NAACL-HLT-Short 2004)*, pp. 101–104, Boston, MA, USA, May 2004.

[Toutanova & Suzuki$^+$ 08] K. Toutanova, H. Suzuki, A. Ruopp. Applying morphology generation models to machine translation. In *46th Annual Meeting of the Association for Computational Linguistics (ACL 2008)*, pp. 514–522, Columbus, OH, USA, June 2008.

[Turian & Ratinov$^+$ 10] J. Turian, L. Ratinov, Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In *48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pp. 384–394, Uppsala, Sweden, July 2010.

[Wang & Lafferty$^+$ 96] Y.-Y. Wang, J. Lafferty, A. Waibel. Word clustering with parallel spoken language corpora. In *4th International Conference on Spoken Language Processing (ICSLP 1996)*, pp. 2364–2367, Philadelphia, USA, October 1996.

[Wuebker & Huck$^+$ 12] J. Wuebker, M. Huck, S. Peitz, M. Nuhn, M. Freitag, J.-T. Peter, S. Mansour, H. Ney. Jane 2: Open source phrase-based and hierarchical statistical machine translation. In *24th International Conference on Computational Linguistics (COLING 2012): Demonstration Papers*, pp. 483–492, Mumbai, India, December 2012.

[Wuebker & Peitz$^+$ 13] J. Wuebker, S. Peitz, F. Rietig, H. Ney. Improving statistical machine translation with word class models. In *2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pp. 1377–1381, Seattle, USA, October 2013.

[Xia & McCord 04] F. Xia, M. McCord. Improving a statistical mt system with automatically learned rewrite patterns. In *20th International Conference on Computational Linguistics (COLING 2004)*, pp. 508–514, Geneva, Switzerland, August 2004.

[Yang & Kirchhoff 06] M. Yang, K. Kirchhoff. Phrase-based backoff models for machine translation of highly inflected languages. In *11th Conference on European Chapter of Association for Computational Linguistics (EACL 2006)*, pp. 3–7, Trento, Italy, April 2006.

[Zens & Och$^+$ 02] R. Zens, F. J. Och, H. Ney. Phrase-based statistical machine translation. In M. Jarke, J. Koehler, G. Lakemeyer, editors, *25th German Conference on Artificial Intelligence (KI2002)*, Vol. 2479 of *Lecture Notes in Artifi-*

cial Intelligence (LNAI), pp. 18–32, Aachen, Germany, September 2002. Springer Verlag.

[Zens 08] R. Zens. *Phrase-based Statistical Machine Translation: Models, Search, Training*. Ph.D. thesis, Department of Computer Science, RWTH Aachen University, Aachen, Germany, February 2008.

[Zhao & Xing$^+$ 05] B. Zhao, E. P. Xing, A. Waibel. Bilingual word spectral clustering for statistical machine translation. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005) Workshop on Building and Using Parallel Texts*, pp. 25–32, Ann Arbor, USA, June 2005.