Morphological Solutions for Arabic Statistical Machine Translation and Sentiment Analysis

by

Mohammad Kassem Salameh

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

Morphologically complex languages such as Arabic pose several challenges in Natural Language Processing (NLP) due to their complexity and token sparsity. Most techniques approach the problem by transforming the words of the language from their sparse surface form representation to a less sparse form before processing. The transformation usually takes the form of a morphological analysis or a morphological segmentation.

This dissertation addresses two tasks in Arabic NLP: Statistical Machine Translation(SMT) and Sentiment Analysis. To improve English-Arabic SMT, we apply segmentation on Arabic to decrease token sparsity and enhance the correspondence between tokens of the English and Arabic language. However, due to this segmentation, the translation system is limited to extracting features based on morphemes (partial words) and only outputting morphemes during decoding. Such a system lacks knowledge of the original form of the words.

We further improve translation from English to Arabic by combining both segmented and desegmented views of the target language. The system can benefit from segmentation's sparsity reduction and verifies its generation of correct words. We present a language-independent technique to desegmentation that approaches the problem as a string transduction task. We propose a new algorithm that desegments the decoder's search space encoded as a lattice, thus allowing the system to use features from the desegmented view of the search space. We extend the phrase-based statistical machine translation system to allow desegmentation during the decoding process on the fly. In addition, we conduct an experimental study to verify what matters most in morphologically segmented SMT models.

Our second task is sentiment analysis, where we resort to Arabic lemmatization to improve sentiment analysis of Arabic tweets and blog posts. We explore translation in the opposite direction, from Arabic into English in order to evaluate the loss of sentiment predictability when Arabic social media posts are translated to English, manually or using an SMT system. We use state-of-the-art Arabic and English sentiment Analysis systems and develop automatically generated Arabic lexicons from lemmatized tweets to improve this task.

Preface

This thesis is an original work by Mohammad Salameh in collaboration with his supervisors Dr. Greg Kondrak and Dr. Colin Cherry. I was responsible for the implementation and experiments between the systems. The writing in the published papers was in collaboration with the co-authors.

Chapter 4 of this thesis has been published as Salameh, M., Cherry, C., and Kondrak, G. (2013). Reversing morphological tokenization in English-to-Arabic SMT. In *Proceedings of the 2013 NAACL HLT Student Research Workshop*, pages 47–53, Atlanta, Georgia.

Chapter 5 of this thesis has been published as Salameh, M., Cherry, C., and Kondrak, G. (2014). Lattice desegmentation for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 100–110. I was responsible for implementing the lattice desegmentation system. I used a decoder and tuning implementations by Dr. Colin Cherry.

Chapter 6 of this thesis has been published as Salameh, M., Cherry, C., and Kondrak, G. (2015a). What matters most in morphologically segmented smt models? *Syntax, Semantics and Structure in Statistical Translation*, page 65.

Chapter 7 of this thesis has been published as Salameh, M., Cherry, C., and Kondrak, G. (2016), Integrating Morphological Desegmentation into Phrase-based Decoding to NAACL HLT, San Diego, California. I was responsible for building the desegmentation system as a feature and integrating it with the SMT system.

Chapter 8 of this thesis has been published as Salameh, M., Mohammad, S., and Kiritchenko, S. (2015b). Sentiment after translation: A case-study on Arabic social media posts. pages 767–777. I was responsible for building an

Arabic sentiment analysis system and conducting the experiments. I used an English sentiment analysis system developed by NRC.

To my parents Sabah and Kassem
for their love, prayers and encouragement

To my wife Dana
for her patience, love and support

And to my lovely daughter Yasmina

وَمِنْ ءَايَـٰتِهِۦ خَلْقُ ٱلسَّمَـٰوَٰتِ وَٱلْأَرْضِ وَٱخْتِلَـٰفُ أَلْسِنَتِكُمْ وَأَلْوَٰنِكُمْ ۚ إِنَّ فِى ذَٰلِكَ
لَءَايَـٰتٍ لِّلْعَـٰلِمِينَ ﴿٢٢﴾

And among His Signs is the creation of the heavens and the earth,
and the difference of your languages and colours.
Verily, in that are indeed signs for men of sound knowledge.

(Quran 30:22)

Acknowledgements

I owe my deepest gratitude to my supervisor Dr. Greg Kondrak. His support, guidance and funding during the last few years enabled me to complete this thesis. He provided me with a broad understanding of NLP through participating in several NLP competitions. I could not have hoped for a better supervisor.

I am truly indebted to my second supervisor Dr. Colin Cherry. I have learned a great deal from his enthusiasm in sharing his expertise and his knowledge in SMT. I would also like to thank him for the opportunity of having an internship at NRC.

I am grateful to Dr. Saif Mohammad and Dr. Svetlana Kiritchenko for introducing me to the area of sentiment analysis. Their critical guidance and helpful comments through our highly collaborative effort on Arabic sentiment analysis allowed me to acquire several skills.

I would like to thank the University of Alberta for the financial support through scholarships and teaching assistantship opportunities. Many thanks to the members of my examining committees: Dr. Osmar Zaiane, Dr. Abrahm Hindle and Dr. Nizar Habash. for their valuable feedback. I would also like to thank my colleague Garrett Nicolai for our fruitful NLP discussions and for proofreading the thesis.

Special thanks to Dr. Nashat Mansour and Dr. Rached Zantout, my Master's advisors at the Lebanese American University for introducing me to the area of Natural Language Processing.

I would to thank Amine Trabelsi, Mohammed Tarek Elmorsy and Hadi Sabaa for being great friends. They made life at the grad school fun and were always available and supportive when in need.

I would like to sincerely thank my Aunt Sukaina Salameh and Uncle Ali Issa for their continuous motivation by providing educational support that encouraged me to pursue my goals. Many thanks to my in-laws Dr. Mahmoud and Zahra Awad for their love and prayers.

I am eternally indebted to my parents Kassem and Sabah, for the sacrifices that they have made on my behalf. I could not have finished this thesis without their support and prayers. I would also like to thank my siblings Mazen, Nour and Farah for their encouragement and motivation.

Finally, I would like to thank my wonderful wife Dana for standing beside me during the past few years. Her love, patience, and understanding through everything we have been through have made this journey an enjoyable and tolerable one. Also, loving thanks to our daughter Yasmina who has brought more joy to our family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Machine Translation: A Brief History

Machine Translation (MT) is defined as the task of translating from one natural language to another. It is one of the oldest subfields in Artificial Intelligence and Natural Language Processing. The story goes back to World War II where machine translation was conceived as a decipherment problem. In 1947, Warren Weaver, a pioneer MT researcher formulated the problem as: given an encoded sentence with strange symbols, the problem is to find best decoded sentence (Weaver, 1955).

There are two main directions in Machine translation: rule-based MT and statistical MT. Rule based MT adopts the use of manually created linguistic rules (Somers, 1992). While manual creation of such rules is expensive and while it is difficult to represent complex morphology and syntax as rules for some rich languages, the recent remarkable progress in MT is due to statistical methods. Statistical Machine Translation (SMT) aims to learn rules automatically from a bilingual corpus. A statistical model is learned from the data by finding cooccurrences between words in source and target languages. The idea of SMT was pioneered by IBM researchers in the late 1980's by presenting the *Candide* SMT system (Brown et al., 1990, 1993) .

The progress in MT led it to be integrated with several applications such as cross-lingual information retrieval and speech translation. Currently, machine translation output is being used as a first draft translation for several translation agencies, where human translators add further adjustments and

fixes (Michael Denkowski and Lavie, 2014). Several SMT systems are freely available online such as Google Translate and Microsoft Bing Translator. They usually provide usable translations between English and some Western European languages.

1.2 Motivation

Translation, similar to most Natural Language Processing tasks, is exacerbated when it involves a morphologically complex language such as Arabic, Finnish and Czech. Such languages present formidable challenges because of their complexity and large number of inflections and derivations of words. Different features such as person, gender, number, tense, etc. are expressed by some modifications to the word. Such modifications can take the form of an affix concatenation or a word derivation that is based on templates. However, all these different forms share the same meaning which is usually expressed by their lemma. Therefore, this leads to data sparsity and poor word representation in any system. Most NLP techniques approach these problem by transforming the words of the morphologically complex language from their surface[1] form coarse-grained representation to a fine-grained form before processing. The transformation usually takes the form of a morphological analysis or a morphological segmentation. This dissertation focuses on providing morphological solutions to two tasks in Arabic NLP: Statistical Machine Translation(SMT) and Sentiment Analysis.

Morphological segmentation is an effective technique for statistical machine translation (SMT) when translating from and to Arabic. When translating from English into Arabic, segmentation on Arabic decreases token sparsity and enhances the correspondence between tokens of the English and Arabic language. Table 1.1 shows an English example with its Arabic translation provided with its *transliteration*[2] that illustrates aspects of translation and its

[1] word form as it originally appeared in the text

[2] We provide Arabic examples with Habash-Saoudi-Buckwalter (Habash et al., 2007) transliteration scheme that maps Arabic characters to ASCII Roman script to enhance readability.

complexity. Notice how each of the Arabic words encapsulates the meaning of several English words. The *segmentation* splits Arabic words into their morphemic representation. Unlike English, prepositions and pronouns in Arabic are encoded as bound morphemes, with the former represented as a prefix and the latter represented as a suffix in لغبطته *lgbTth* "to his eminence" . Such properties exacerbate the SMT task through an increase of Arabic token sparsity and a large out-of-vocabulary rate. However, due to this segmentation, the translation system is limited to extracting features based on morphemes (partial words) and only outputting morphemes during decoding. Such a system lacks knowledge of the original form of the words. Our goal in this thesis is to improve translation from English into Arabic. We would like the SMT system to output morphologically and orthographically correct Arabic words while also benefiting from segmentation. Also, we aim to allow our SMT models to capture several aspects related to Arabic morphology and present a fluent translation.

On the other hand, lemmatization (as a form of morphological analysis) for morphologically complex languages is a practical approach for sparsity reduction. When applied on Arabic social media text for a sentiment analysis task, lemma forms maintain the main aspects of the meaning of their surface forms while their clitical and inflectional features are dropped. Such an approach shows significant improvements when sentiment analysis systems are trained on lemmas compared to training on the original surface forms.

However, lemmatization alone might not suffice as a solution due to the use of dialectal Arabic terms in social media. Current Arabic morphological analyzers find difficulty in lemmatizing such terms because they lack strict writing standards. Also, available lexicons have low coverage to terms appearing on social media. We aim to create a state-of-the art Arabic sentiment analysis system that relies on automatically generated Arabic lexicon from tweets. Also, we want to benefit from both available English sentiment analysis resources and advances in Arabic-English SMT to measure the loss of sentiment predictability when Arabic social media posts are translated into English manually and automatically.

English	we explained the matter to his eminence .							
Arabic	واوضحنا الامر لغبطته .							
Transliteration	wAwDHnA			AlAmr	lgbTth			.
Segmentation	w+	AwDH	+nA	Al+ Amr	l+	gbTħ	+h	.
Gloss	and	explained	we	the matter	to	eminence	his	.

Table 1.1: English example with its Arabic translation.

1.3 Approaches

In this section, we provide a summary of the approaches we propose to English-Arabic SMT and sentiment analysis, with the different challenges faced and the main contributions in this thesis.

In Chapter 4, we address the desegmentation task that appears as part of the SMT pipeline when translating into morphologically segmented Arabic. Arabic segmentation appears as a preprocessing step in an SMT pipeline (Figure 1.1). The segmentation process applies adjustment rules on Arabic to regularize the segmented substrings; hence simply concatenating the segmented forms might not result in the original Arabic word ($lgbTth \rightarrow l+\ gbTħ+h$). An SMT system trained on segmented Arabic outputs Arabic translations in segmented form, which has to be desegmented again to allow readability and evaluation. Our first challenge is to address the the current desegmentation techniques and their limitations. Then, we provide a novel desegmentation technique that handles the problem through a string transduction approach. Our technique overcomes limitations of current techniques by learning desegmentation rules automatically, and at the same time memorizing long sequence mappings.

Our technique showed gains and near-perfect desegmentation when applied on naturally occurring segmented Arabic words. But when we applied it to Arabic SMT output, it did not make much difference compared to other techniques. This comes as a result of applying desegmentation as a post-processing step in the SMT pipeline (Figure 1.1). Desegmentation merely operates on the final output of the SMT system and does not contribute to how the output is generated. Hence, decoding errors propagated to the desegmentation step are

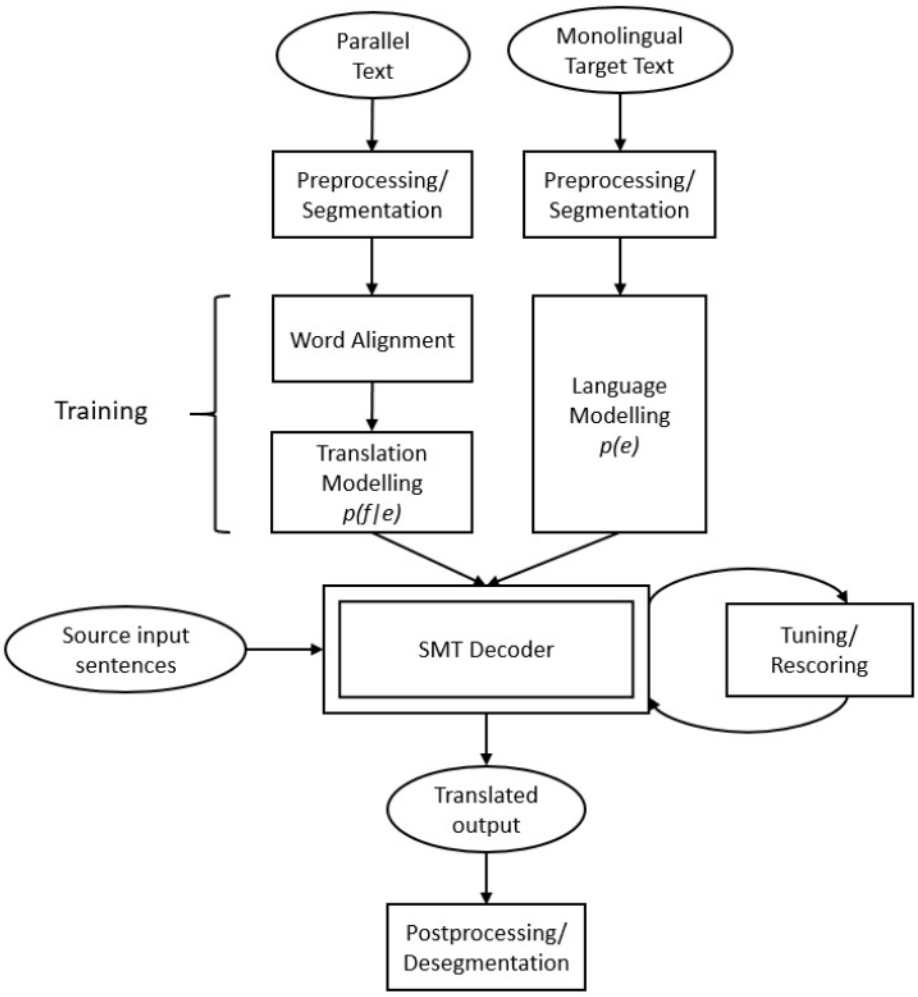


Figure 1.1: Statistical Machine Translation pipeline involving target language segmentation as a preprocessing step.

unlikely to be fixed.

In Chapter 5, we address this challenge by providing a solution to integrating desegmentation in the SMT process, rather than keeping it as a post-processing step. Instead of segmenting the final output, we desegment the search space (represented as a lattice) that is built from segmented Arabic tokens. Thus, the system is provided with two views of its search space: a segmented and a desegmented view. This enables extraction of features from desegmented words such as scoring using an unsegmented language model. As a consequence, our system benefits from the sparsity reduction of morphological segmentation, and at the same time, outputs correct desegmented Arabic

words.

Our lattice desegmentation system shows significant improvement on English-Arabic and English-Finnish translations compared to systems applying desegmentation on the final output. Yet, we were faced with several questions and challenges from experts and reviewers regarding our approach:

1. Is it possible to get the same gains by desegmenting a different SMT component such as the phrase-table rather than desegmenting the lattice?

2. Is it possible to run an unsegmented model, where segmentation only influences the word alignment and also get the same improvement in translation quality?

These thoughtful questions gave us a chance to go further and evaluate other desegmentation options and confirm whether there are better options than desegmenting the lattice.

In Chapter 6, we provide a systematic exploration of a space of possible solutions for translating directly into unsegmented Arabic text while still obtaining the benefits of segmentation. Segmentation provides a potential boost to many components of the SMT system such as the alignment model, translation phrase table, and language models. We illustrate which component benefits the most from segmentation by providing scenarios that differ in where desegmentation is applied.

Our experiments on English-Arabic translation attribute the benefits of segmentation to *phrases with flexible boundaries*, which give the SMT system the freedom of generating words that can span multiple phrases. Also, we show that the use of an unsegmented language model contributes to the BLEU score but discourages the use of morphologically decomposable words. Also, our lattice desegmentation approach proved to be the best desegmentation, while in the other explored approaches we are giving up this important property of phrases with flexible boundaries. At this point, we are ready to move the desegmentation process directly into the decoder.

In Chapter 7, we present an algorithm that extends the phrase-based SMT system by enabling desegmentation while decoding. Sequences of morphemes are desegmented on the fly, followed by word-level feature extraction from desegmented forms. This goal is far more challenging; yet, we get a model that is easier to use and more accessible to the SMT community than the lattice desegmentation system.

Finally in Chapter 8, we investigate a task that is only possible due to advances made in Arabic-English SMT. Sentiment Analysis has predominantly been on English. Thus there exist many sentiment resources for English, but less so for other languages such as Arabic. But with improvements in statistical machine translation systems over the last decade, we no longer have to rely on strictly monolingual sentiment analysis systems. An alternative is to translate the focus language text into English, and run an English sentiment analysis system.

We provide an evaluation on the loss in sentiment predictability when Arabic blog posts and tweets are translated into English manually and automatically. In the process, we create a state-of-the-art Arabic sentiment analysis system and compare its performance to an English one trained on Arabic translations. We discover that even though translation significantly reduces the human ability to recover sentiment, automatic sentiment systems are still able to capture sentiment information from the translations. we conduct qualitative and quantitative studies to investigate why we observe these results. We find that sentiment expressions are often mistranslated into neutral expressions when translated. Further, mistranslation of ambiguous words, sarcasm, metaphoric expressions, and incorrect word-reordering are common reasons why translations fail to preserve sentiment. In the process, we also create a state-of-the-art Arabic sentiment analysis system and automatically generated Arabic lexicon from tweets.

1.4 Thesis Organization

We now describe the organization of the thesis. In Chapter 2, we provide a brief overview on phrase-based statistical machine translation system and its different components. In Chapter 3, we provide an overview on the Arabic language, its morphology, syntax, and main challenges. In addition, we discuss the basic approaches to handle Arabic in Statistical Machine Translation. Chapters 4 through 8 provide details on our approaches for English-Arabic SMT and sentiment analysis, as outlined in the previous section. Finally, we conclude by summarizing our contributions in Chapter 9 and discussing directions for future work.

Chapter 2

Phrase-based Statistical Machine Translation

This section provides a short introduction to Statistical Machine Translation (SMT). It covers main approaches for training an SMT system, tuning, decoding, as well as methods for system evaluation. It includes definitions for the main terminologies used in SMT context, that will be referenced in the subsequent chapters.

2.1 The Standard Model

Machine translation is a well defined problem in NLP that involves translating from a *source* language to a *target* language. Given a source sentence **f**, we want to find the best translation **e** in target language among a set of candidate translations E. The problem can be represented as

$$e^* = \arg\max_{e \in E} p(e|f) \tag{2.1}$$

Currently, the phrase-based SMT model is considered the best performing model for several languages. Unlike the word-based model, the unit of translation is a phrase. A **phrase** in SMT is merely a sequence of one or more words and does not necessarily correspond to the linguistic meaning.

In SMT, we want to learn translation rules statistically from a parallel corpus. A **parallel corpus** is a large set of translation examples that are sentence aligned. When translating, we want to guarantee two main criteria

in the output: (1) the output is fluent, and (2) it also conveys the meaning of the source sentence by using words which are correct translations. Fluency is measured using a **Language Model** (LM) which gives high probability score for an output sentence e that is more fluent than other sentences. The LM parameters are estimated based on a target language monolingual corpus, which is abundantly available. Adequacy is measured using a **Translation Model** (TM) which gives a conditional probability score for any pair f and e. The TM parameters are estimated by training on a bilingual parallel corpus. Hence, the model is further decomposed into smaller components using the noisy channel approach of the generative model:

$$e^* = \arg\max_{e \in E} \overbrace{p(f|e)}^{TranslationModel} \times \overbrace{p(e)}^{LanguageModel} \tag{2.2}$$

Due to syntactic differences between languages, translated phrases have to be reordered. A **Reordering model** learns to penalize larger skips for consecutive phrases using a distortion limit parameter. This leads to phrases being translated in a monotonic, swapped, or discontinuous manner with respect to the neighboring phrase.

2.2 Language Model

A language model evaluates whether a sequence of words is fluent. It defines a probability distribution that assigns a high probability to a fluent sequence of words and a low probability to an unlikely sequence. Assuming we are translating into English, a LM should assign a high probability to *"exports to china increase"* and a low one to *"increase to exports china"* since it has an incorrect grammar and word order.

Given a word sequence $W = w_1, w_2, ..., w_{|W|}$, the **n-gram language model**[1] uses chain rule to calculate the probability of W, denoted as $P(W)$, as a product of conditional probabilities of individual words given their history of preceding words:

[1] An n-gram is an n-token sequence of words

$$
\begin{aligned}
p(W) &= p(w_1) \times p(w_2|w_1) \times p(w_3|w_1w_2) \times p(w_4|w_1w_2w_3) \times ... \times p(w_n|w_1...w_{n-1}) \\
&= \prod_{i=1}^{|W|}(w_i|w_1...w_{i-1})
\end{aligned}
\tag{2.3}
$$

The product is further simplified using a conditional independence Markov assumption that takes the history of each word up to n words.

$$
p(W) \approx \prod_{i=1}^{|W|} p(w_i|w_{i-n+1}^{i-1}) \tag{2.4}
$$

For a trigram language model that uses a history of two preceding words, the probability of *"he lives in canada"* is estimated as:

$$
\begin{aligned}
p(he\ lives\ in\ canada) = p(he|<s>) &\times p(lives|<s>\ he) \times p(in|he\ lives) \\
&\times p(canada|lives\ in) \times p(<\backslash s>|in\ canada)
\end{aligned}
$$

where the symbols $<s>$ and $<\backslash s>$ corresponds to start-of-sentence and end-of-sentence respectively.

The probabilities are estimated from a large monolingual English corpus using Maximum Likelihood Estimation. For a trigram language model,

$$
p(w_k|w_iw_j) \approx \frac{count(w_iw_jw_k)}{\sum_w count(w_iw_jw)} \tag{2.5}
$$

where *count(X)* returns the number of instances of the sequence X in the corpus. During testing, several n-grams might not be seen in the training data set. In order to account for previously unseen n-grams, **smoothing** techniques are used to move some probability mass from seen n-grams to unseen ones.

2.3 Translation Model

As mentioned earlier, a parallel corpus is needed for training a translation model $p(f|e)$. Phrase extraction for the TM depends on a word aligned parallel text. An **alignment** is a mapping between a word in source text to its correspondence in target language text. Alignment is learned automatically through an unsupervised approach from the statistics of the parallel text by

measuring how frequent a source word co-occur with target word. Given the generated alignments, we can extract phrases $(\bar{f}, \bar{e})$ such that they are consistent with the word alignment. This means that words in $\bar{f}$ has to be aligned with words in $\bar{e}$ and vice versa. Also the words in the extracted phrase have to be continuous. Figure 2.1 represents an alignment matrix with alignment points between an English sentence and its Arabic translation. The alignment is used to extract phrases shown in the adjacent table for source segments of length three or less. Notice that no phrase is extracted to *wwzEt mydAlyAt* due to discontinuity on the source. Also, notice that the word *tqdyr* has no alignment with the source, but still we can extract phases that includes this word as long as there is no discontinuity in the extracted pair.

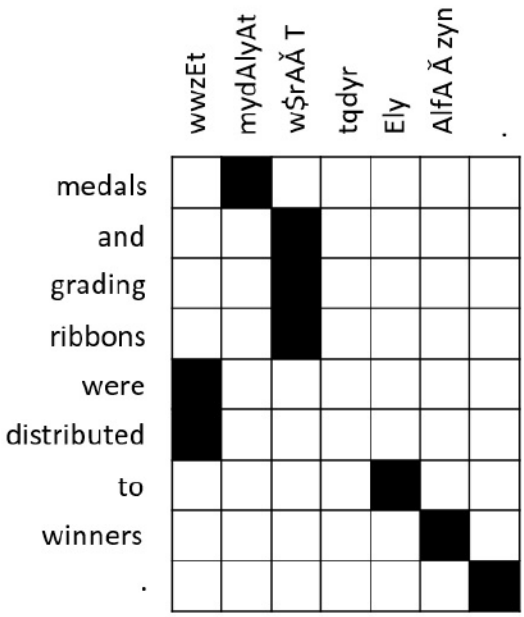
Phrase extraction allows building a **phrase table** that consists of source phrase, target phrase and four probability scores estimated from the world aligned parallel corpus. The scores are: inverse phrase translation probability $\phi(f|e)$, inverse lexical weighting $lex(f|e)$, direct phrase translation probability $\phi(e|f)$, and direct lexical weighting $lex(e|f)$. The translation probability is calculated as

$$\phi(\bar{f}|\bar{e}) = \frac{count(\bar{e}, \bar{f})}{\sum_{\bar{f}_i} count(\bar{e}, \bar{f}_i)} \tag{2.6}$$

where $count(\bar{e}, \bar{f})$ returns the number of times phrases $\bar{e}$ and $\bar{f}$ co-occur in the parallel corpus. The lexical weights checks how well words in $\bar{f}$ and $\bar{e}$ translate to each other. Each word's lexical translation probability can be determined by counting the words aligned to it in our training corpus. Given these lexical probabilities, the lexical weight of a phrase pair can be calculated as the product of lexical probability scores for the aligned words in the phrase.

2.4 Decoding

Given the above models, decoding in SMT involves finding the translation with the highest probability (Equation 1). The decoding task is considered to be an NP-complete problem. Solutions provided are heuristic and might not guarantee finding an optimal translation. The phrase-based SMT system uses a **multi-stack beam search decoder** that work as follows. First, source



English	Arabic
medals	mydAlyAt
and grading ribbons	wšrAŷT
and grading ribbons	wšrAŷT tqdyr
to	Ely
to	tqdyr Ely
were distributed	wwzEt
winners	AlfAŷzyn
winners .	AlfAŷzyn .
to winners	Ely AlfAŷzyn
to winners	tqdyr Ely AlfAŷzyn
to winners .	Ely AlfAŷzyn .
to winners .	tqdyr Ely AlfAŷzyn .
.	.

Figure 2.1: Alignment matrix between an English sentence "medals and grading ribbons were distributed to winners" and its Arabic translation ووزعت ميداليات وشرائط تقدير على الفائزين, where the black squares represent an alignment point between source and target words. The adjacent table shows extracted phrases with source-length less than or equal to three.

segments with different sizes are picked up from the source sentence, and their translation options are retrieved from the phrase table. The search space can be constructed by considering different permutations of the source segments and their multiple translation options. But this can lead to an exponential growth of the search space. The decoder handles this by initializing n **stacks** (priority queues), where n is the source-length. Then, the decoder starts constructing the output from left to right, making sure it covers all the words in the source sentence such that that no input word is covered twice. Partial translations considered during this search are called **hypotheses**. The process of appending current hypothesis with new translations for an uncovered source segment is called **hypothesis expansion**. As hypotheses get expanded, they are allocated to the appropriate stack, such that stack-i accepts hypotheses covering i source words. The purpose of the stack is to keep hypotheses sorted according to their partial scores, and discard the ones that cost more than the best hypothesis according to a threshold value. The process is known

as **threshold pruning**. Another pruning approach is to keep the top k hypotheses in the stack by restricting the stack size to k. The approach is called **histogram pruning**. Pruning reduces the complexity of the model and search space, and restricts the hypothesis expansion to a certain limit, thus allowing faster decoding.

Each partial hypothesis is represented through a **state** which saves several pieces of information used to calculate the partial cost. The state contains all of the information needed to score an expansion of its partial hypothesis. This includes the source coverage vector, the n-gram language model context and any information needed to calculate reordering probabilities. When two hypotheses have identical states, they can be recombined. The reason is that the better scoring hypothesis will always be considered in the path to the final hypothesis. As a result, the decoder can benefits from **hypothesis recombination** through having an efficient search by not considering paths with high cost.

As we explained, pruning is based on comparing partial scores of hypotheses covering same number of source words. The problem is that some source phrases are easier to translate than the others, thus making the their partial hypotheses scores incomparable. Given the sentence "italy topped the list of cotton importing countries", it is easier to translate "the list of" compared to translating "cotton importing countries". The phrase "the list of" has more common words that can provide its translation with a cheaper score compared to "cotton importing countries". Such unfair score comparison might lead to hypothesis of "cotton importing countries" be pruned. The problem is addressed by adding a **future cost** to the partial score of the hypothesis. The purpose of future cost is to estimate how difficult it is to translate the rest of the sentence, given the partial hypothesis.

2.5 Parameter Tuning

The models that we have described (translation model, language model and reordering model) can be represented as features $h_i(x)$ in a log-linear model

that are weighted based on their significance in generating best possible translations.

$$p(x) \propto \exp \sum_{i=1}^{n} \lambda_i h_i(x) \tag{2.7}$$

The translation score generated from the combination of these components is a weighted score based on parameters λ_i.

$$x_{best}(\lambda_1, ..., \lambda_n) = \arg\max_x \sum_{i=1}^{n} \lambda_i h_i(x) \tag{2.8}$$

Parameter tuning refers to setting the parameter weights λ_i so that the translation quality is optimized. Optimization is usually based on the BLEU score, which is an automatic evaluation metric (described in Section 2.6). The well-known approaches for tuning are Minimum Error Rate Training (MERT) and Margin Infused Relaxed Algorithm (MIRA). Both of these approaches run on a development set that is decoded and conduct optimization on the n-best list or lattice generated from the development set. Given an initial weight setting, **MERT** optimizes the parameters iteratively by updating the weights and re-decoding to expand n-best list or lattice. It is not feasible when the number of features is more than 30. **MIRA** is an online learning algorithm that updates the weights with respect to a loss function (calculated based on BLEU score) and marginal constraints. The updates are minimal while obtaining a margin larger than the loss of incorrect classification. Unlike MERT, MIRA can train an SMT system with millions of features.

2.6 Evaluation Metrics

Human evaluation is always expensive and time-consuming. As a result, a trusted automatic evaluation metrics that can evaluate the output's fluency and adequacy have always been in demand. Several automatic evaluation metrics are available for the machine translation task. They all evaluate the SMT output based on a **reference**, which is a human-generated translation for the test sentence. The performance of these metrics is always debatable.

But given two translation outputs from 2 different SMT systems, they can indicate to some extent which of the two is better. The most popular metric is the Bilingual Evaluation Understudy score known as **BLEU**(Papineni et al., 2002). It calculates the geometric mean of precision based on matching n-grams between the SMT output and one or more references. Since it is a metric based on precision, the tendency to prefer shorter translation is controlled by a brevity-penalty when the length of the reference is larger than the length of the output. It is defined as :

$$BLEU\text{-}4 = brevityPenalty \times exp(\frac{1}{4}\sum_{i=1}^{4} log\ precision_i) \tag{2.9}$$

where

$$precision_i = \frac{number\ of\ matching\ i\text{-}grams}{total\ number\ of\ i\text{-}grams\ in\ the\ output}$$

$$brevityPenalty = min(1, \frac{output\text{-}length}{reference\text{-}length})$$

Another metric is the Word Error Rate. It uses a dynamic programming approach to calculate the number of edits needed to transform the output translation into the reference. The metric is more concerned about the sequence of words in comparison with the reference. It is not frequently used for SMT as it places hard penalty on reordering even if the translation is correct. We will use this metric to evaluate the desegmentation of a word (Chapter 4), where reordering is not a concern. It is defined as

$$WER = \frac{substitutions\ +\ deletions\ +\ insertions}{reference\text{-}length}$$

Translation Error Rate (TER) (Snover et al., 2006) is similar to WER and calculates the number of edits required to change the SMT output into the reference. But it also considers the block movement, which is a sequence of words (called *shifts*), among the editing steps. We use this metric in Chapter5 to evaluate the translation quality.

Chapter 3

Arabic Language and Challenges

This section briefly describes the orthographic, morphological, and syntactic characteristics of the Arabic language that exacerbate the process of SMT and sentiment analysis. We also illustrate the challenges faced in English-Arabic SMT along with the various approaches presented in recent papers to tackle them.

3.1 Arabic Orthography

In this section, we describe main Arabic orthography issues that cause computational complexity during translation. Unlike English, Arabic text is written from right to left. Also, the text is either fully diacritized, partially diacritized or undiacritized. **Diacritics** are small marks that are added to consonants. They help disambiguate any semantic differences and usually change based on syntactic conditions. For example, diacritizing the word كتب *ktb* (undiacritized form) can disambiguate its meaning as either: كَتَبَ *kataba* "wrote", كُتُب *kutub* "books", or كُتِبَ *kutiba* "has been written" (where *a, u,* and *i* are diacritics). Also, several Arabic letters are spelled inconsistently using different forms. The Alif has different variants such as bare Alif ا *A*, hamzated Alif أ *Â* إ *Ǎ* and Alif Mada آ *Ā* . The letter ي Ya is sometimes written as a dottless Ya ى.

Such inconsistent spelling of Arabic characters and use of characters interchangeably lead to word sparsity, ambiguity, and poor probabilistic estimations of words. These challenges are mainly addressed by applying orthographic nor-

malization schemes. An initial preprocessing step in Arabic SMT is to remove all diacritics. Although this might add more ambiguity for some words, it has a positive influence on the quality of translation as it decreases sparsity. Another preprocessing step is to normalize Arabic script by converting different forms of Alif إ أ آ ا and Ya ي ى to bare Alif ا and dotless Ya ى respectively.

El Kholy and Habash (2012a) studied the effect of orthographical normalization on Arabic by experimenting on reduced (with Alif and Ya normalization) and enriched Arabic (with predicting right Alif and Ya format). Better results were shown with the reduced Arabic set. Currently, most of the research in En-Ar uses reduced Arabic format. Furthermore, Arabic SMT output is compared to an undiacritized script during evaluation.

3.2 Arabic Morphological Complexity

Another challenge that contributes to the Arabic complexity is morphology. The richness in Arabic morphology and the large number of Arabic word forms increase the sparsity and out-of-vocabulary word rate in the corpus. Arabic words are derived based on a root and pattern. For example, the word كاتِب *kAtib* "writer" is derived from the root كَتَب *k-t-b* "wrote" and pattern *1A2i3*[1], where 1, 2 and 3 are the consonants in the roots respectively. Verbs in Arabic inflect for aspect, mood, voice, and subject (person, number and gender), while nouns inflect for gender, number, state and case. Each of these features has its own set of subcategories which results in different inflected word forms. Although templatic morphology presents interesting challenges, we will not address it in this thesis.

Each inflected word form (base) can be attached to several optional clitics. A general form of Arabic word can be represented by:

```
[question+ [conjunction+ [particle+ [determiner+ BASE +pronoun]]]]
```
(3.1)

where each of these clitical categories has a fairly large set of clitics.

As a result of such morphological richness, a lemma can have thousands

[1] 1A2i3 is an abstract template for nouns that means *the one that enacts the action in the root*

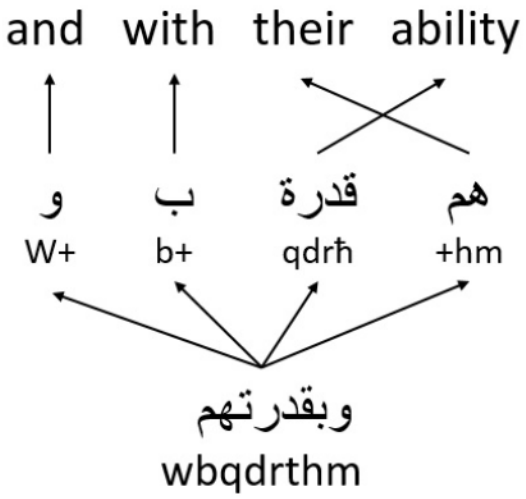


Figure 3.1: Segmentation of Arabic word وبقدرتهم *wbqdrthm* and its alignment with its English translation.

of inflected forms. El Kholy and Habash (2012a) show that the number of Arabic **tokens**[2] in a bilingual corpus is estimated to be less by 20% to the number of English tokens. However, the number of Arabic word **types**[3] in the same corpus is as twice as the number of English word types. In addition, alignment gets complicated between the English source and Arabic, as a single Arabic word can correspond to multiple English words spread at distinct places in the sentence. Notice that the clitical categories (question, conjunction, particle, determiner and pronoun) represented as bound morphemes in Arabic, correspond to separate words in English. The next two subsections illustrate how to improve correspondence and decrease token sparsity between English and Arabic.

3.2.1 Segmentation

Segmentation is the process of splitting a word into consecutive substrings. Several NLP tasks rely on segmentation as a method for decreasing token sparsity, and using a simplified form of a word that can still encompass some aspects of its meaning, especially when dealing with morphologically complex languages.

[2]number of running words in a corpus
[3]number of distinct words in a corpus

In this thesis, we define **Morphological Segmentation** as a process that involves both:

- splitting the word into consecutive substrings
- applying morphological and orthographic adjustments on required substrings

We adopt the same definition proposed by Habash (2010) for tokenization. We use the the term "segmentation" instead of "tokenization" in order not to confuse it with the tokenization task used in statistical machine translation context and limited to dividing text into sequence of words by separating punctuation marks Throughout this thesis, we use the above definition of segmentation on Arabic language, unless stated otherwise.

As mentioned in this section, Arabic words can be formed by attaching clitics to an inflected base form. The attachment is not a simple concatenation step; rather, it involves several character transformations on morpheme boundaries that might cause the word to be different from its individual parts. During segmentation, the **orthographic adjustments** step applies character transformations to convert the segmented substrings back to their basic form. For example, the Arabic word لطفلته *lTflth* "for his child" is segmented as ه+ طفلة +ل *l+ Tfl*ħ*+h* "for child his". The orthographic adjustment transforms letter "t" to "ħ"; otherwise, we get the incorrect Arabic word طفلت *Tflt*.

Segmenting Arabic words into correct morphemes is a challenging task. Checking whether a sequence of characters is part of the stem or constitute a clitic requires morphological analysis of the word. After detecting the morphemes in a word, the next task is to choose at what clitical point to segment and how to group the tokens. Habash et al. (2009) provide Morphological Analysis and Disambiguation for Arabic tool (MADA) which can perform word analysis and apply different segmentation schemes. MADA uses Support Vector Machine to select the best word analysis from a list of analyses provided by Buckwalter Arabic Morphological Analyzer (BAMA)(Buckwalter, 2004).

Arabic	وستقدم خدمتها بالتعاون معهم			
Buckwalter	wstqdm	xdmthA	bAltςAwn	mςhm
Translation	and she will offer	her service	in collaboration	with them
D1	w+ stqdm	xdmthA	bAltςAwn	mςhm
D2	w+ s+ tqdm	xdmthA	b+ AltςAwn	mςhm
D3	w+ s+ tqdm	xdmp +hA	b+ Al+ tςAwn	mς +hm
S2	w+s+ tqdm	xdmp +hA	b+Al+ tςAwn	mς +hm
ATB	w+ s+ tqdm	xdmp +hA	b+ AltςAwn	mς +hm

Table 3.1: Different segmentation schemes presented in literature for Arabic. We refer the reader to Habash (2010) for a complete list of schemes.

The choice of the level of granularity in segmentation usually depends on the task. A certain segmentation scheme, that works well for Information Retrieval, might complicate the process in Machine Translation, and vice versa. Several segmentation schemes were proposed in literature for Arabic. Each of the segmentation schemes differ in where segmentation is applied. Table 3.2.1 shows commonly used segmentation schemes for Arabic, where the segmentation point is decided based on the general form of Arabic word. These schemes were presented by Sadat and Habash (2006), except for **S2** that was presented by Badr et al. (2008)

- **D1:** segments *question* and *conjunction* clitics
- **D2:** segments *question*, *conjunction* and *particle* clitics
- **D3:** segments all clitics
- **S2:** segments all clitics, but groups enclitics(question, conjunction, particle and determiner) together as one token.
- **ATB:** the Penn Arabic Treebank segmentation scheme segments all clitics except for determiner proclitic ال *Al* which is left attached to the *base form.*

We adopt the Penn Arabic Treebank (PATB) segmentation scheme in all of our experiments in this dissertation. El Kholy and Habash (2012a) show in

an experimental study that it has the best impact on English-Arabic SMT compared to other schemes.

3.2.2 Desegmentation

When translating from Arabic into English, the segmentation is a form of preprocessing, and the output translation is readable, space-separated English. However, when translating from English to Arabic, the output will be in a segmented form, which cannot be compared to the original unsegmented reference. Simply concatenating the segmented morphemes cannot fully reverse this process, because of character transformations that occurred during segmentation. Hence, when translating into a segmented language, the segmentation must be reversed to make the generated text readable and evaluable.

Desegmentation is the process of converting the segmented form of words into their original orthographically and morphologically correct surface form. This includes concatenating tokens into complete words and reversing any character transformations that may have taken place. The task is considered challenging since the process can result with multiple desegmented options.

For example, the following three words شراؤه *šrAŵh,* شراءه *šrAÂh*, and شرائه *šrAŷh* meaning "its purchase" differ in their Hamza form (ؤ ئ ء ŵ ŷ Â) as they inflect for nominative, accusative and genitive case respectively. When segmented, they share the same segmentation form شراء+ه *šrA' +h*. Without any knowledge of the context of شراء+ه *šrA' +h*, or the diacritic mark appearing on the Hamza or the character preceeding it(which are usually dropped when Arabic text is processed), it will be difficult to know its original unsegmented from.

The segmentation and desegmentation operations can place a word in three different states when performed in any NLP task. To resolve any ambiguity in definitions(in later chapters), we define the word states as:

- **Segmented Word**: is a word that is morphologically segmented and is presented in its segmented form
- **Unsegmented word** is a word in its original form and was never seg-

mented. We also use the term **surface form** to denote a word in its unsegmented form.

- **Desegmented Word**: is a word that went through a segmentation process, and was then desegmented. Notice that the process of desegmentation does not always result with the original word form before segmentation i.e its unsegmented word form

The techniques proposed for the desegmentation task fall into three categories: simple, rule-based and table-based (Badr et al., 2008). We devote Chapter 4 to discuss these approaches in detail with their limitations.

3.3 Syntactic Challenges

Word order in Arabic has some degree of freedom, thus affected by the syntactic relations represented within the complex morphology. Arabic sentences can have different word order, but some cases appear more than others. Arabic has both nominal sentences (starts with a noun) and verbal sentences, but the verbal sentences are more frequent. In machine translation, this will have an influence on the fluency when translating into Arabic, as sentences in English are mainly nominal (except for imperative sentences). Adjectives in Arabic follow the noun they modify and the *idafa* noun phrase. *Idafa* construct is similar to the English possessive and compound nouns. For example, "the student's notebook" is translated to مفكرة الطالب "notebook the-student" in Arabic. In Subject-Verb-Object sentences, the verb agrees with the subject in gender and number. In Verb-subject-object order sentences, the verb only agrees in gender. Also the verb subjects in Arabic can be dropped (the reader usually infers it from the verb conjugation).

Phrase-based SMT has some limitations in modeling the syntactical relationship since phrases in SMT are generated by the alignment and not by a syntactic parser. This results in limitations in handling long distance agreement appearing in the selected phrases by the decoder. Such long distance dependencies are affected by the differences between syntax structure in En-

glish and Arabic.

3.4 Summary

In this chapter, we highlighted the main problems pertaining to the complexity of the Arabic language in NLP tasks. We discussed the main challenges related to orthography, morphology and syntax. In this thesis, we do not try to solve challenges related to Arabic syntax. We mainly concentrate on providing solutions related to morphology in the context of SMT and sentiment analysis.

Chapter 4

Reversing Morphological Segmentation in English-to-Arabic SMT

In the previous chapter, we discussed "segmentation" as a solution that overcomes the morphological complexity of the Arabic language. The advantages of Arabic morphological segmentation in SMT appears as a reduction in lexical sparsity and improved correspondence between English and Arabic tokens. Nevertheless, the output of such system is segmented and unreadable Arabic. Recombining segmented tokens to generate original word forms is not a trivial task, due to morphological, phonological and orthographic adjustments that occur during segmentation. In this chapter, we address "desegmentation" as an indispensable process in the SMT pipeline which handles this problem. We review a number of desegmentation schemes for Arabic, such as rule-based and table-based approaches and show their limitations. We then propose a novel desegmentation scheme that uses a character-level discriminative string transducer to predict the original form of a segmented word. In a comparison to a state-of-the-art approach by El Kholy and Habash (2012a), we demonstrate slightly better desegmentation error rates without the need for any handcrafted rules. We also demonstrate the effectiveness of our approach in an English-to-Arabic translation task.

4.1 Introduction

Statistical machine translation (SMT) relies on segmentation to split sentences into meaningful units for easy processing. For morphologically complex languages, such as Arabic or Turkish, this may involve splitting words into morphemes. Throughout this thesis, we adopt the definition of tokenization proposed by Habash (2010), which incorporates both morphological segmentation as well as orthographic character transformations (unless stated otherwise). To use an English example, the word *tries* would be morphologically segmented as *"try + s"*, which involves orthographic changes at morpheme boundaries to match the lexical form of each token. When translating into a segmented language, the segmentation must be reversed to make the generated text readable and evaluable. Desegmentation is the process of converting tokenized words into their original orthographically and morphologically correct surface form. This includes concatenating tokens into complete words and reversing any character transformations that may have taken place.

For languages like Arabic, segmentation can facilitate SMT by reducing lexical sparsity. Figure 4.1 shows how the morphological segmentation of the Arabic word وسيمنعهم "and he will prevent them" simplifies the correspondence between Arabic and English tokens, which in turn can improve the quality of word alignment, rule extraction and decoding. When translating from Arabic into English, the segmentation is a form of preprocessing, and the output translation is readable, space-separated English. However, when translating from English to Arabic, the output will be in a segmented form, which cannot be compared to the original reference without desegmentation. Simply concatenating the segmented morphemes cannot fully reverse this process, because of character transformations that occurred during segmentation.

The techniques that have been proposed for the desegmentation task fall into three categories (Badr et al., 2008). The simplest desegmentation approach concatenates morphemes based on token markers without any adjustment. Table-based desegmentation maps segmented words into their surface form with a look-up table built by observing the segmenter's input and out-

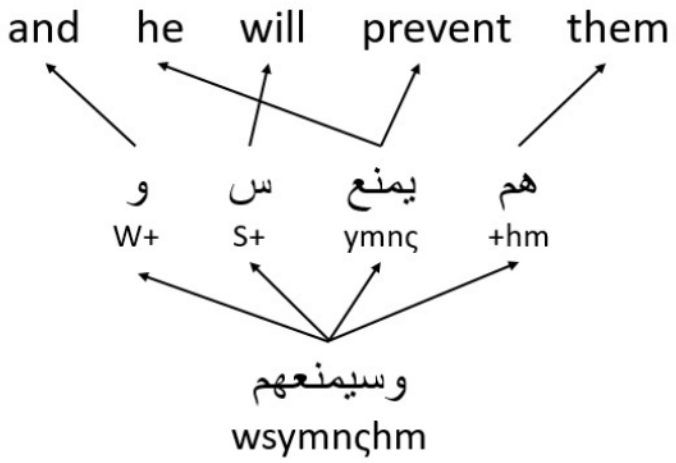


Figure 4.1: Alignment between tokenized form of *"wsymnçhm"* وسيمنعهم and its English translation.

put on large amounts of text. Rule-based desegmentation relies on hand-built rules or regular expressions to convert the segmented form into the original surface form. Other techniques use combinations of these approaches. Each approach has its limitations: rule-based approaches are language specific and brittle, while table-based approaches fail to deal with sequences outside of their tables.

We present a new detokenization approach that applies a discriminative sequence model to predict the original form of the tokenized word. Like table-based approaches, our sequence model requires large amounts of tokenizer input-output pairs; but instead of building a table, we use these pairs as training data. By using features that consider large windows of within-word input context, we are able to intelligently transition between rule-like and table-like behavior.

Our experimental results on Arabic text demonstrate an improvement in terms of sentence error rate[1] of 11.9 points over a rule-based approach, and 1.1 points over a table-based approach that backs off to rules. More importantly, we achieve a slight improvement over the state-of-the-art approach of El Kholy and Habash (2012a), which combines rules and tables, using a 5-gram word-based language model to disambiguate conflicting table entries. In addition,

[1]Sentence Error rate is the percentage of sentences containing at least one error after desegmentation.

our desegmentation method results in a small BLEU improvement over a rule-based approach when applied to English-to-Arabic SMT.

4.2 Related Work

Sadat and Habash (2006) address the issue of lexical sparsity by presenting different preprocessing schemes for Arabic to English SMT. The schemes include simple segmentation, orthographic normalization, and decliticization. The combination of these schemes results in improved translation output. This is one of many studies on normalization and segmentation for translation from Arabic, which we will not attempt to review completely here.

Badr et al. (2008) show that segmenting Arabic also has a positive influence on English to Arabic SMT. They apply two segmentation schemes on Arabic text and introduce desegmentation schemes through a rule-based approach, a table-based approach, and a combination of both. The combination approach desegments words first using the table, falling back on rules for sequences not found in the table.

El Kholy and Habash (2012a) extend Badr's work by presenting a larger number of segmentation and desegmentation schemes and comparing their effects on SMT. They introduce an additional desegmentation schemes based on the SRILM *disambig* utility (Stolcke, 2002), which utilizes a 5-gram unsegmented language model to decide among different alternatives found in the table. They test their schemes on naturally occurring Arabic text and SMT output. Their newly introduced desegmentation scheme outperforms the rule-based and table-based approaches introduced by Badr et al. (2008), establishing the current state-of-the-art.

4.2.1 Desegmentation Schemes in Detail

Rule-based desegmentation involves manually defining a set of transformation rules to convert a sequence of segmented tokens into their surface form. For example, the noun "*llrŷys*" للرئيس "to the president" is tokenized as *"l+ Alrŷys"* (*l+* "to" *Alrŷys* "the president") in the PATB tokenization scheme. Note

Rule	Segmented	Desegmented
l+Al → ll	l+ Alrŷys	llrŷys
ħ+(pron) → t(pron)	Abnħ+hA	AbnthA
y+(pron) → A(pron)	Alqy+h	AlqAh
'+(pron) → ŷ	AntmA'+hm	AntmAŷhm
y+y → y	ςyny+y	ςyny
n+n → n	mn+nA	mnA
mn+m → mm	mn+mA	mmA
ςn+m → ςm	ςn+mA	ςmA
An+lA → AlA	An+lA	AlA

Table 4.1: Desegmentation rules of El Kholy and Habash (2012a), with examples. *pron* stands for pronominal clitic

that the definite article "*Al*" ال is kept attached to the noun. In this case, detokenization requires a character-level transformation after concatenation, which we can generalize using the rule:

$$l{+}Al \rightarrow ll.$$

Table 4.1 shows the rules provided by El Kholy and Habash (2012a), which we employ throughout this chapter.

There are two principal problems with the rule-based approach. First, rules fail to account for unusual cases. For example, the above rule mishandles cases where "*Al*" ال is a basic part of the stem and not the definite article *"the"*. Thus, 'l+ AlςAb' (*l+* "to" *AlςAb* "games") is erroneously desegmented to *llEAb* للعاب instead of the correct form is "*lAlςAb*" لالعاب. Second, rules may fail to handle sequences produced by segmentation errors. For example, the word "*bslTħ*" بسلطة "with power" can be erroneously segmented as *"b+slT+h"*, while the correct segmentation is *"b+slTħ"*. The erroneous segmentation will be incorrectly desegmented as *"bslTh"*.

The table-based approach memorizes mappings between words and their segmented form. Such a table is easily constructed by running the segmenter on a large amount of Arabic text, and observing the input and output. The desegmentation process consults this table to retrieve the unsegmented surface forms of segmented words. In the case where a segmented word has several observed surface forms, the most frequent form is selected. This approach fails

when the sequence of segmented words is not in the table. In morphologically complex languages like Arabic, an inflected base word can attract many optional clitics, and tables may not include all different forms and inflections of a word.

The SRILM-disambig scheme introduced by El Kholy and Habash (2012a) extends the table based approach to use an unsegmented Arabic language model to disambiguate among the different alternatives. Hence, this scheme can make context-dependent desegmentation decisions, rather than always producing the most frequent unsegmented surface form. Both the SRILM-disambig scheme and the table-based scheme have the option to fall back on either rules or simple concatenation for sequences missing from the table.

4.3 Desegmentation as String Transduction

We propose to approach desegmentation as a string transduction task. A string transducer maps between two sets of character symbols. In this task, the transducer maps from characters of the segmented form to characters of the desegmented form. We train a discriminative transducer on a set of segmented-unsegmented word pairs. The set of pairs is initially aligned on the character level, and the alignment pairs become the operations that are applied during transduction. For desegmentation, most operations simply copy over characters, but more complex rules such as $l+\ Al \rightarrow ll$ are learned from the training data as well.

The tool that we use to perform the transduction is DIRECTL+, a discriminative, character-level string transducer, which was originally designed for letter-to-phoneme conversion (Jiampojamarn et al., 2008). To align the characters in each training example, DIRECTL+ uses an EM-based M2M-ALIGNER (Jiampojamarn et al., 2007). After alignment is complete, MIRA training repeatedly decodes the training set to tune the features that determine when each operation should be applied. The features include both character-based n-gram source context and HMM-style target transitions. DIRECTL+ employs a fully discriminative decoder to learn character transformations and

when they should be applied. The decoder resembles a monotone phrase-based SMT decoder, but is built to allow for hundreds of thousands of features.

The following example illustrates how string transduction applies to desegmentation. The segmented and surface forms of *bbrAçthm* ببراعتهم "with their skill" constitute a training instance:

$$b+\_\ brA\varsigma\hbar\_\ +hm \rightarrow bbrA\varsigma thm$$

The instance is aligned during the training phase as:

b+	_b	r	A	ς	ħ_	+	h	m
|	|	|	|	|	|	|	|	|
b	b	r	A	ς	t	ϵ	h	m

The underscore "_" indicates a space, while "ϵ" denotes an empty string. The following operations are extracted from the alignment:

- b+ → b
- A → A
- + → ϵ
- _b → b
- ς → ς
- h → h
- r → r
- ħ_ → t
- m → m

During training, weights are assigned to features that associate operations with context. In our running example, the weight assigned to the b+ → b operation accounts for the operation itself, for the fact that the operation appears at the beginning of a word, and for the fact that it is followed by an underscore; in fact, we employ a context window of 5 characters to the left or right of the source substring "b+", creating a feature for each character-based n-gram in that window.

Modeling the segmentation problem as string transduction has several advantages. The approach is completely language-independent. The context-sensitive rules are learned automatically from examples, without human intervention. The rules and features can be represented in a more compact way than the full mapping table required by table-based approaches, while still elegantly handling words that were not seen during training. Also, since the training data is generalized more efficiently than in simple memorization of

complete segmented-unsegmented pairs, less training data should be needed to achieve good accuracy.

4.4 Experiments

This section presents two experiments that evaluate the effect of the desegmentation schemes on both naturally occurring Arabic and SMT output.

4.4.1 Data

To build our data-driven desegmenters, we use the Arabic part of 4 Arabic-English parallel datasets from the Linguistic Data Consortium as training data. The data sets are: Arabic News (LDC2004T17), eTIRR (LDC2004E72), English translation of Arabic Treebank (LDC2005E46), and Ummah (LDC2004T18). The training data has 107K sentences. The Arabic part of the training data constitutes around 2.8 million words, 3.3 million tokens after segmentation, and 122K word types after filtering punctuation marks, Latin words and numbers (Refer to Table 4.2 for detailed counts).

For training the SMT system's translation and re-ordering models, we use the same 4 datasets from LDC. We also use 200 Million words from LDC Arabic Gigaword corpus (LDC2011T11) to generate a 5-gram segmented/unsegmented language model using SRILM toolkit (Stolcke, 2002).

We use NIST MT 2004 evaluation set for tuning (1075 sentences), and NIST MT 2005 evaluations set for testing (1056 sentences). Both MT04 and MT05 have multiple English references in order to evaluate Arabic to English translation. As we are translating into Arabic, we take the first English translation to be our source in each case. We also use the Arabic halves of MT04 and MT05 as development and test sets for our experiments on naturally occurring Arabic. The segmented Arabic is our input, with the original Arabic as our gold-standard desegmentation.

The Arabic text of the training, development, testing set and language model are all segmented using MADA 3.2 (Habash et al., 2009) with the Penn Arabic Treebank segmentation scheme. The English text in the parallel corpus

is lower-cased and segmented in the traditional sense to strip punctuation marks.

4.4.2 Experimental Setup

To train the desegmentation systems, we generate a table of mappings from segmented forms to surface forms based on the Arabic part of our 4 parallel datasets, giving us complete coverage of the output vocabulary of our SMT system. In the table-based approached, if a segmented form is mapped to more than one surface form, we use the most frequent surface form. For out-of-table words, we fall back on concatenation (in T) or rules (in T+R). For SRILM-Disambig desegmentation, we maintain ambiguous table entries along with their frequencies, and we introduce a 5-gram language model to disambiguate desegmentation choices in context. Like the table-based approaches, the Disambig approach can back off to either simple concatenation (T+LM) or rules (T+R+LM) for missing entries. The latter is a re-implementation of the state-of-the-art system presented by El Kholy and Habash (2012a).

We train our discriminative string transducer using word types from the 4 LDC catalogs. We use M2M-ALIGNER to generate a 2-to-1 character alignments between segmented forms and surface forms. For the decoder, we set Markov order to one, joint n-gram features to 5, n-gram size to 11, and context size to 5. This means the decoder can an utilize contexts up to 11 characters long, allowing it to effectively memorize many words. We found these settings using grid search on the development set, NIST MT04.

For the SMT experiment, we use GIZA++ for the alignment between English and segmented Arabic, and perform the translation using Moses phrase-based SMT system (Hoang et al., 2007), with a maximum phrase length of 5. We apply each desegmentation scheme on the SMT segmented Arabic output test set, and evaluate using the BLEU score (Papineni et al., 2002).

4.4.3 Results

Table 4.3 shows the performance of several desegmentation schemes. For evaluation, we use the sentence and word error rates on naturally occurring Arabic

Data set	Before	After
training set	122,720	61,943
MT04	8,201	2,542
MT05	7,719	2,429

Table 4.2: Type counts by before and after segmentation.

Desegmentation	WER	SER	BLEU
Baseline	1.710	34.3	26.30
Rules(R)	0.590	14.0	28.32
Table(T)	0.192	4.9	28.54
Table+Rules(T+R)	0.122	3.2	28.55
Disambig(T+LM)	0.164	4.1	28.53
Disambig(T+R+LM)	0.094	2.4	28.54
DirecTL+	0.087	2.1	28.55
Disambig+DirecTL+	0.038	1.0	28.56

Table 4.3: Word and sentence error rate of desegmentation schemes on the Arabic reference text of NIST MT05. BLEU score refers to the English-Arabic SMT output.

text, and BLEU score on segmented Arabic output of the SMT system. The baseline scheme, which is a simple concatenation of morphemes, introduces errors in over a third of all sentences. The table-based approach outperforms the rule-based approach, indicating that there are frequent exceptions to the rules in Table 1 that require memorization. Their combination (T+R) fares better, leveraging the strengths of both approaches. The addition of SRILM-Disambig produces further improvements as it uses a language model context to disambiguate the correct desegmented word form. Our system outperforms SRILM-Disambig by a very slight margin, indicating that the two systems are roughly equal. This is interesting, as it is able to do so by using only features derived from the segmented word itself; unlike SRILM-Disambig, it has no access to the surrounding words to inform its decisions. Furthermore, we integrate our system with SRILM-Disambig, such that it disambiguates between the *n-best* desegmentation options that DirecTL+ outputs. The integration results with the lowest error rate due to an added benefit from access to the context of the desegmented word. This level of performance is

achieved without any manually constructed rules.

Improvements in desegmentation do contribute to the BLEU score of our SMT system, but only to a point. Table 4.3 shows three tiers of performance, with no desegmentation being the worst, the rules being better, and the various data-driven approaches performing best. After WER dips below 0.2, further improvements seem to no longer affect SMT quality. Note that BLEU scores are much lower overall than one would expect for the translation in the reverse direction, because of the morphological complexity of Arabic, and the use of one (as opposed to four) references for evaluation.

4.4.4 Analysis

The sentence error rate of 2.1 represents only 21 errors that our approach makes. Among those 21, 11 errors are caused by changing ħ to *h* and vice versa. This is due to writing *p* and *h* interchangeably. For example, *"AjmAly+h"* was desegmented as *"AjmAlyħ"* اجمالية instead of "AjmAlyh" اجماليه. Another 4 errors are caused because of the lack of diacritization, which affects the choice of the Hamza form. For example,*"bnAŵh"* بناؤه, *"bnAŷh"* بنائه and *"bnA'h"* بناءه ("its building") are 3 different forms of the same word where the choice of Hamza ء is dependent on the its diacritical mark or the mark of the character that precedes it. Another 3 errors are attributed to the case of the nominal which it inflects for. The case is affected by the context of the noun which DIRECTL+ has no access to. For example, *"mfkry+hm"* ("thinkers/Dual-Accusative") was desegmented as *"mfkrAhm"* مفكراهم (Dual-Nominative) instead of "mfkryhm" مفكريهم. The last 3 errors are special cases of *"An +y"* which can be desegmented correctly as either *"Any"* اني or *"Anny"* انني.

The table-based desegmentation scheme fails in 54 cases. Among these instances, 44 cases are not in the mapping table, hence resolving back to simple concatenation ended with an error. Our transduction approach succeeds in desegmenting 42 cases out of the 54. The majority of these cases involves changing ħ to *h* and vice versa and changing *l+Al* to *ll*. The only 2 instances where the segmented word is in the mapping table but DIRECTL+ incorrectly

desegments it are due to Hamza case and ħto h case described above. There are 4 instances of the same word/case where both the table scheme and DIRECTL+ fail due to error of segmentation by MADA, where the proper name *qwh* قوه is erroneously segmented as *qw* +ħ. This shows that DIRECTL+ handles the OOV correctly.

The Disambig(T+R+LM) erroneously desegments 27 instances, where 21 out of them are correctly segmented by DIRECTL+. Most of the errors are due to the Hamza and ħ to *h* reasons. It seems that even with a large size language model, the SRILM utility needs a large mapping table to perform well. Only 4 instances were erroneously desegmented by both Disambig and DIRECTL+ due to Hamza and the case of the nominal.

The analysis shows that using small size training data, DIRECTL+ can achieve slightly better accuracy than SRILM scheme. The limitations of using table and rules are handled with DIRECTL+ as it is able to memorize more rules.

4.5 Summary

In this chapter, we addressed the desegmentation problem for Arabic using DIRECTL+, a discriminative training model for string transduction. Our system performs the best among the available systems. It manages to solve problems caused by limitations of table-based and rule-based systems. This allows us to surpass the performance of the SRILM-disambig approach without using hand-crafted rules.

Chapter 5

Lattice Desegmentation for Statistical Machine Translation

In the previous chapter, we addressed desegmentation as a post-processing step in the SMT pipeline that desegments the 1-best-output from the decoder. We provided a solution that address the task as a string transduction problem. Our technique resulted with a near perfect desegmentation on naturally occurring Arabic text, while no improvement is shown on Arabic SMT output. Desegmentation as a post-processing technique could not overcome errors propagated by the decoder. In this chapter, we aim to benefit from desegmentation by integrating it into the SMT process. Instead of desegmenting the 1-best-output, we expand our translation options by desegmenting n-best lists or lattices. We provide a novel lattice desegmentation algorithm that effectively combines both segmented and desegmented views of the target language for a large subspace of possible translation outputs, which allows for inclusion of features related to the desegmentation process, as well as an unsegmented language model (LM). We investigate this technique in the context of English-to-Arabic and English-to-Finnish translation, showing significant improvements in translation quality over desegmentation of 1-best decoder outputs.

5.1 Introduction

Morphological segmentation is considered to be indispensable when translating between English and morphologically complex languages such as Arabic. Morphological complexity leads to much higher type to token ratios than English, which can create sparsity problems during translation model estimation. Morphological segmentation addresses this issue by splitting surface forms into meaningful morphemes, while also performing orthographic transformations to further reduce sparsity. For example, the Arabic noun للدول *lldwl* "to the countries" is segmented as *l+* "to" *Aldwl* "the countries". When translating from Arabic, this segmentation process is performed as input preprocessing and is otherwise transparent to the translation system. However, when translating into Arabic, the decoder produces segmented output, which must be **desegmented** to produce readable text. For example, *l+ Aldwl* must be converted to *lldwl*.

Desegmentation is typically performed as a post-processing step that is independent from the decoding process. While this division of labor is useful, the pipeline approach may prevent the desegmenter from recovering from errors made by the decoder. Despite the efforts of the decoder's various component models, the system may produce mismatching segments, such as *s+ hzymp*, which pairs the future particle *s+* "will" with a noun *hzymp* "defeat", instead of a verb. In this scenario, there is no right desegmentation; the post-processor has been dealt a losing hand.

In this chapter, we show that it is possible to maintain the sparsity-reducing benefit of segmentation while translating directly into unsegmented text. We desegment a large set of possible decoder outputs by processing n-best lists or lattices, which allows us to consider both the segmented and desegmented output before locking in the decoder's decision. We demonstrate that significant improvements in translation quality can be achieved by training a linear model to re-rank this transformed translation space.

5.2 Related Work

Translating into morphologically complex languages is a challenging and interesting task that has received much recent attention. Most techniques approach the problem by transforming the target language in some manner before training the translation model. They differ in what transformations are performed and at what stage they are reversed. The transformation might take the form of a morphological analysis or a morphological segmentation.

5.2.1 Morphological Analysis

Many languages have access to morphological analyzers, which annotate surface forms with their lemmas and morphological features. Bojar (2007) incorporates such analyses into a factored model, to either include a language model over target morphological tags, or model the generation of morphological features. Other approaches train an SMT system to predict lemmas instead of surface forms, and then inflect the SMT output as a post-processing step (Minkov et al., 2007; Clifton and Sarkar, 2011; Fraser et al., 2012; El Kholy and Habash, 2012b). Alternatively, one can reparameterize existing phrase tables as exponential models, so that translation probabilities account for source context and morphological features (Jeong et al., 2010; Subotin, 2011). Of these approaches, ours is most similar to the translate-then-inflect approach, except we translate and then desegment. In particular, Toutanova et al. (2008) inflect and re-rank n-best lists in a similar manner to how we desegment and re-rank n-best lists or lattices.

5.2.2 Morphological Segmentation

Instead of producing an abstract feature layer, morphological segmentation transforms the target sentence by segmenting relevant morphemes, which are then handled as regular tokens during alignment and translation. This is done to reduce sparsity and to improve correspondence with the source language (usually English). Such a segmentation can be produced as a byproduct of analysis (Oflazer and Durgar El-Kahlout, 2007; Badr et al., 2008; El Kholy

and Habash, 2012a), or may be produced using an unsupervised morphological segmenter such as Morfessor (Luong et al., 2010; Clifton and Sarkar, 2011). Work on target language morphological segmentation for SMT can be divided into three subproblems: segmentation, desegmentation and integration. This chapter is concerned primarily with the integration problem, but we will discuss each subproblem in turn.

The usefulness of a target segmentation depends on its correspondence to the source language. If a morphological feature does not manifest itself as a separate token in the source, then it may be best to leave its corresponding segment attached to the stem. A number of studies have looked into what granularity of segmentation is best suited for a particular language pair (Oflazer and Durgar El-Kahlout, 2007; Badr et al., 2008; Clifton and Sarkar, 2011; El Kholy and Habash, 2012a). Since our focus here is on integrating segmentation into the decoding process, we simply adopt the segmentation strategies recommended by previous work: the Penn Arabic Treebank scheme for English-Arabic (El Kholy and Habash, 2012a), and an unsupervised scheme for English-Finnish (Clifton and Sarkar, 2011).

Desegmentation is the process of converting segmented words into their original surface form. For many segmentations, especially unsupervised ones, this amounts to simple concatenation. However, more complex segmentations, such as the Arabic tokenization provided by MADA (Habash et al., 2009), require further orthographic adjustments to reverse normalizations performed during segmentation. Badr et al. (2008) present two Arabic desegmentation schemes: table-based and rule-based. El Kholy and Habash (2012a) provide an extensive study on the influence of segmentation and desegmentation on English-to-Arabic SMT. They introduce an additional desegmentation technique that augments the table-based approach with an unsegmented language model. A literature review on the desegmentation techniques and their limitations has been covered in the previous chapter (section 4.2.1). Also, we propose a discriminatively-trained character transducer that replaces rule-based desegmentation (section 4.3). In this chapter and the next two chapters (6 and 7), we adopt the Table+Rules approach of El Kholy and Habash (2012a)

for English-Arabic, while concatenation is sufficient for English-Finnish.

Work on integration attempts to improve SMT performance for morphologically complex target languages by going beyond simple pre- and post-processing. Oflazer and Durgar El-Kahlout (2007) desegment 1000-best lists for English-to-Turkish translation to enable scoring with an unsegmented language model. Unlike our work, they *replace* the segmented language model with the unsegmented one, allowing them to tune the linear model parameters by hand. We use both segmented and unsegmented language models, and tune automatically to optimize BLEU.

Like us, Luong et al. (2010) tune on unsegmented references,[1] and translate with both segmented and unsegmented language models for English-to-Finnish translation. However, they adopt a scheme of word-boundary-aware morpheme-level phrase extraction, meaning that target phrases include only complete words, though those words are segmented into morphemes (this approach will be revisited in the next chapter). This enables full decoder integration, where we do n-best and lattice re-ranking. But it also comes at a substantial cost: when target phrases include only complete words, the system can only generate word forms that were seen during training. In this setting, the sparsity reduction from segmentation helps word alignment and target language modeling, but it does not result in a more expressive translation model. Furthermore, it becomes substantially more difficult to have non-adjacent source tokens contribute morphemes to a single target word. For example, when translating "with his blue car" into the Arabic بسيارته الزرقاء *bsyArth AlzrqA'*, the target word *bsyArth* is composed of three tokens: *b+* "with", *syArp* "car" and *+h* "his". With word-boundary-aware phrase extraction, a phrase pair containing all of "with his blue car" must have been seen in the parallel data to translate the phrase correctly at test time. With lattice desegmentation, we need only to have seen *AlzrqA'* "blue" and the three morphological pieces of *bsyArth* for the decoder and desegmenter to assemble

[1]Tuning on unsegmented references does not require substantial modifications to the standard SMT pipeline. For example, Badr et al. (2008) also tune on unsegmented references by simply desegmenting SMT output before MERT collects sufficient statistics for BLEU.

the phrase.

5.3 Methods

Our goal in this work is to benefit from the sparsity-reducing properties of morphological segmentation while simultaneously allowing the system to reason about the final surface forms of the target language. We approach this problem by augmenting an SMT system built over target segments with features that reflect the desegmented target words. In this section, we describe our various strategies for desegmenting the SMT system's output space, along with the features that we add to take advantage of this desegmented view.

5.3.1 Baselines

The two obvious baseline approaches each decode using one view of the target language. The **unsegmented** approach translates without segmenting the target. This trivially allows for an unsegmented language model and never makes desegmentation errors. However, it suffers from data sparsity and poor token-to-token correspondence with the source language.

The **one-best desegmentation** approach segments the target language at training time and then desegments the one-best output in post-processing. This resolves the sparsity issue, but does not allow the decoder to take into account features of the desegmented target. To the best of our knowledge, we are the first group to go beyond one-best desegmentation for English-to-Arabic translation. In English-to-Finnish, although alternative integration strategies have seen some success (Luong et al., 2010), the current state-of-the-art performs one-best-desegmentation (Clifton and Sarkar, 2011).

5.3.2 n-best Desegmentation

The one-best approach can be extended easily by desegmenting n-best lists of segmented decoder output. Doing so enables the inclusion of an unsegmented target language model, and with a small amount of bookkeeping, it also allows the inclusion of features related to the operations performed during deseg-

mentation (see Section 5.3.4). With new features reflecting the desegmented output, we can re-tune our enhanced linear model on a development set. Following previous work, we will desegment 1000-best lists (Oflazer and Durgar El-Kahlout, 2007).

Once n-best lists have been desegmented, we can tune on unsegmented references as a side-benefit. This could improve translation quality, as it brings our training scenario closer to our test scenario (test BLEU is always measured on unsegmented references). In particular, it could address issues with translation length mismatch. Previous work that has tuned on unsegmented references has reported mixed results (Badr et al., 2008; Luong et al., 2010).

5.3.3 Lattice Desegmentation

An n-best list reflects a tiny portion of a decoder's search space, typically fixed at 1000 hypotheses. Lattices[2] can represent an exponential number of hypotheses in a compact structure. In this section, we discuss how a lattice from a multi-stack phrase-based decoder such as Moses (Koehn et al., 2007) can be desegmented to enable word-level features.

Finite State Analogy

A phrase-based decoder produces its output from left to right, with each operation appending the translation of a source phrase to a growing target hypothesis. Translation continues until each source word has been covered exactly once (Koehn et al., 2003).

The search graph of a phrase-based decoder can be interpreted as a lattice, which can be interpreted as a finite state acceptor over target strings. In its most natural form, such an acceptor emits target phrases on each edge, but it can easily be transformed into a form with one edge per token, as shown in Figure 5.1a. This is sometimes referred to as a word graph (Ueffing et al., 2002), although in our case the segmented phrase table also produces tokens that correspond to morphemes.

[2]Or forests for hierarchical and syntactic decoders.

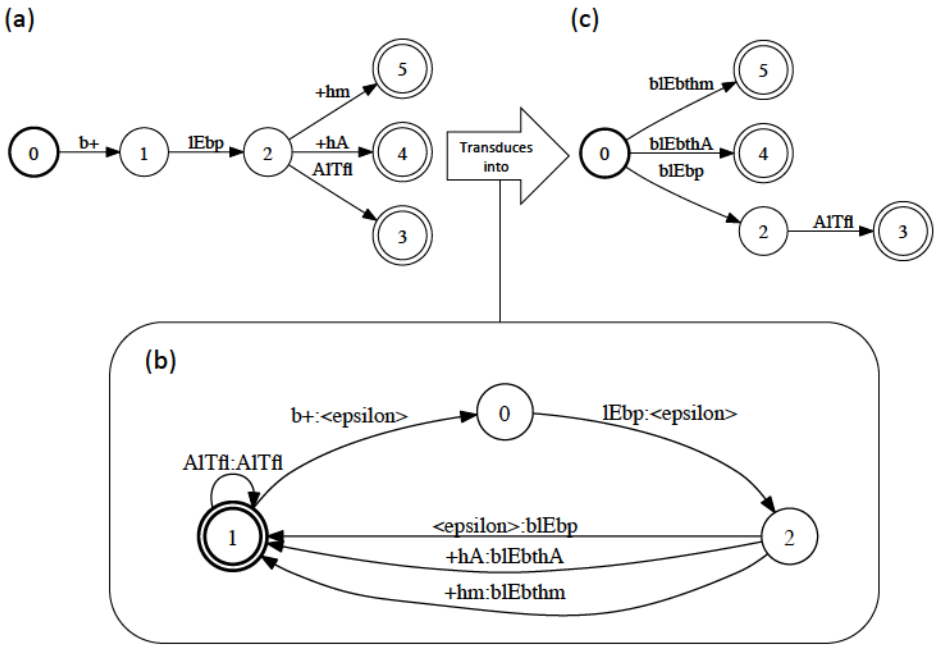


Figure 5.1: The finite state pipeline for a lattice translating the English fragment "with the child's game". The input morpheme lattice (a) is desegmented by composing it with the desegmenting transducer (b) to produce the word lattice (c). The tokens in (a) are: *b+* "with", *lEbp* "game", *+hm* "their", *+hA* "her", and *AlTfl* "the child".

Our goal is to desegment the decoder's output lattice, and in doing so, gain access to a compact, desegmented view of a large portion of the translation search space. This can be accomplished by composing the lattice with a **desegmenting transducer** that consumes morphemes and outputs desegmented words. This transducer must be able to consume every word in our lattice's output vocabulary. We define a word using the following regular expression:

$$\text{[prefix]* [stem] [suffix]* | [prefix]+ [suffix]+} \tag{5.1}$$

where [prefix], [stem] and [suffix] are non-overlapping sets of morphemes, whose members are easily determined using the segmenter's segment boundary markers.[3] The second disjunct of Equation 5.1 covers words that have no clear stem, such as the Arabic لـه *lh* "for him", segmented as *l+* "for" *+h* "him". Equation 5.1 may need to be modified for other languages or segmentation

[3]Throughout this thesis, we use "+" to mark morphemes as prefixes or suffixes, as in *w+* or *+h*. In Equation 5.1 only, we overload "+" as the Kleene cross: $X+ == XX*$.

schemes, but our techniques generalize to any definition that can be written as a regular expression.

A desegmenting transducer can be constructed by first encoding our desegmenter as a table that maps morpheme sequences to words. Regardless of whether the original desegmenter was based on concatenation, rules or table-lookup, it can be encoded as a lattice-specific table by applying it to an enumeration of all words found in the lattice. We can then transform that table into a finite state transducer with one path per table entry. Finally, we take the closure of this transducer, so that the resulting machine can transduce any sequence of words. The desegmenting transducer for our running example is shown in Figure 5.1b. Note that tokens requiring no desegmentation simply emit themselves. The lattice (Figure 5.1a) can then be desegmented by composing it with the transducer (5.1b), producing a desegmented lattice (5.1c). This is a natural place to introduce features that describe the desegmentation process, such as scores provided by a desegmentation table, which can be incorporated into the desegmenting transducer's edge weights.

We now have a desegmented lattice, but it has not been annotated with an unsegmented (word-level) language model. In order to annotate lattice edges with an n-gram LM, every path coming into a node must end with the same sequence of $(n-1)$ tokens. If this property does not hold, then nodes must be split until it does.[4] This property is maintained by the decoder's recombination rules for the segmented LM, but it is not guaranteed for the desegmented LM. Indeed, the expanded word-level context is one of the main benefits of incorporating a word-level LM. Fortunately, LM annotation as well as any necessary lattice modifications can be performed simultaneously by composing the desegmented lattice with a finite state acceptor encoding the LM (Roark et al., 2011).

In summary, we are given a segmented lattice, which encodes the decoder's translation space as an acceptor over morphemes. We compose this acceptor with a desegmenting transducer, and then with an unsegmented LM acceptor,

[4]Or the LM composition can be done dynamically, effectively decoding the lattice with a beam or cube-pruned search (Huang and Chiang, 2007).

producing a fully annotated, desegmented lattice. Instead of using a tool kit such as OpenFst (Allauzen et al., 2007), we implement both the desegmenting transducer and the LM acceptor programmatically. This eliminates the need to construct intermediate machines, such as the lattice-specific desegmenter in Figure 5.1b, and facilitates working with edges annotated with feature vectors as opposed to single weights.

Programmatic Desegmentation

Lattice desegmentation is a non-local lattice transformation. That is, the morphemes forming a word might span several edges, making desegmentation non-trivial. Luong et al. (2010) address this problem by forcing the decoder's phrase table to respect word boundaries, guaranteeing that each desegmentable token sequence is local to an edge. Inspired by the use of non-local features in forest decoding (Huang, 2008), we present an algorithm to find **chains** of edges that correspond to desegmentable token sequences, allowing lattice desegmentation with no phrase-table restrictions. This algorithm can be seen as implicitly constructing a customized desegmenting transducer and composing it with the input lattice on the fly.

Before describing the algorithm, we define some notation. An input morpheme lattice is a triple $\langle n_s, \mathcal{N}, \mathcal{E} \rangle$, where $\mathcal{N}$ is a set of nodes, $\mathcal{E}$ is a set of edges, and $n_s \in \mathcal{N}$ is the start node that begins each path through the lattice. Each edge $e \in \mathcal{E}$ is a 4-tuple $\langle \mathit{from}, \mathit{to}, \mathit{lex}, w \rangle$, where from, $\mathit{to} \in \mathcal{N}$ are head and tail nodes, lex is a single token accepted by this edge, and w is the (potentially vector-valued) edge weight. Tokens are drawn from one of three non-overlapping morpho-syntactic sets: $\mathit{lex} \in \mathit{Prefix} \cup \mathit{Stem} \cup \mathit{Suffix}$, where tokens that do not require desegmentation, such as complete words, punctuation and numbers, are considered to be in Stem. It is also useful to consider the set of all outgoing edges for a node $n.\mathit{out} = \{e \in \mathcal{E} | e.\mathit{from} = n\}$.

With this notation in place, we can define a **chain** c to be a sequence of edges $[e_1 \dots e_l]$ such that for $1 \leq i < l : e_i.\mathit{to} = e_{i+1}.\mathit{from}$. We denote singleton chains with $[e]$, and when unambiguous, we abbreviate longer chains with their start and end node $[e_1.\mathit{from} \rightarrow e_l.\mathit{to}]$. A chain is **valid** if it emits the beginning

of a word as defined by the regular expression in Equation 5.1. A valid chain is **complete** if its edges form an entire word, and if it is part of a path through the lattice that consists only of words. In Figure 5.1a, the complete chains are $[0 \rightarrow 2]$, $[0 \rightarrow 4]$, $[0 \rightarrow 5]$, and $[2 \rightarrow 3]$. The path restriction on complete chains forces words to be bounded by other words in order to be complete.[5] For example, if we removed the edge $2 \rightarrow 3$ (*AlTfl*) from Figure 5.1a, then $[0 \rightarrow 2]$ ([*b+ lEbp*]) would cease to be a complete chain, but it would still be a valid chain. Note that in the finite-state analogy, the path restriction is implicit in the composition operation.

Algorithm 1 desegments a lattice by finding all complete chains and replacing each one with a single edge. It maintains a work list of nodes that lie on the boundary between words, and for each node on this list, it launches a depth first search to find all complete chains extending from it. The search recognizes the valid chain c to be complete by finding an edge e such that $c+e$ forms a chain, but not a valid one. By inspection of Equation 5.1, this can only happen when a prefix or stem follows a stem or suffix, which always marks a word boundary. The chains found by this search are desegmented and then added to the output lattice as edges. The nodes at end points of these chains are added to the work list, as they lie at word boundaries by definition. Note that although this algorithm creates completely new edges, the resulting node set $\mathcal{N}'$ will be a subset of the input node set $\mathcal{N}$. The complement $\mathcal{N} - \mathcal{N}'$ will consist of nodes that are word-internal in all paths through the input lattice, such as node 1 in Figure 5.1a.

Programmatic LM Integration

Programmatic composition of a lattice with an n-gram LM acceptor is a well understood problem. We use a dynamic program to enumerate all $(n-1)$-word contexts leading into a node, and then split the node into multiple copies, one for each context. With each node corresponding to a single LM context, annotation of outgoing edges with n-gram LM scores is straightforward.

[5] Sentence-initial suffix morphemes and sentence-final prefix morphemes represent a special case that we omit for the sake of brevity. Lacking stems, they are left segmented.

Algorithm 1 Desegment a lattice $\langle n_s, \mathcal{N}, \mathcal{E} \rangle$

{Initialize output lattice and work list WL}
$n'_s = n_s$, $\mathcal{N}' = \emptyset$, $\mathcal{E}' = \emptyset$, $WL = [n_s]$
while $n = WL$.pop() **do**
  {Work on each node only once}
  if $n \in \mathcal{N}'$ **then continue**
  $\mathcal{N}' = \mathcal{N}' \cup \{n\}$
  {Initialize the chain stack $\mathcal{C}$}
  $\mathcal{C} = \emptyset$
  for $e \in n.out$ **do**
    if $[e]$ is valid **then** $\mathcal{C}$.push($[e]$)
  {Depth-first search for complete chains}
  while $[e_1, \ldots, e_l] = \mathcal{C}.pop()$ **do**
    {Attempt to extend chain}
    for $e \in e_l.to.out$ **do**
      if $[e_1 \ldots e_l, e]$ is valid **then**
        $\mathcal{C}.push([e_1, \ldots, e_l, e])$
      else
        Mark $[e_1, \ldots, e_l]$ as complete
    {Desegment complete chains}
    if $[e_1, \ldots, e_l]$ is complete **then**
      WL.push($e_l.to$)
      $\mathcal{E}' = \mathcal{E}' \cup \{\text{deseg}([e_1, \ldots, e_l])\}$
return $\langle n'_s, \mathcal{N}', \mathcal{E}' \rangle$

5.3.4 Desegmentation Features

Our re-ranker has access to all of the features used by the decoder, in addition to a number of features enabled by desegmentation.

Desegmentation Score We use a table-based desegmentation method for Arabic, which is based on segmenting an Arabic training corpus and memorizing the observed transformations to reverse them later. Finnish does not require a table, as all words can be desegmented with simple concatenation. The Arabic table consists of $X \rightarrow Y$ entries, where X is a target morpheme sequence and Y is a desegmented surface form. Several entries may share the same X, resulting in multiple desegmentation options. For the sake of symmetry with the unambiguous Finnish case, we augment Arabic n-best lists

or lattices with only the most frequent desegmentation Y.[6] We provide the desegmentation score $\log p(Y|X) = \log\left(\frac{\text{count of } X \rightarrow Y}{\text{count of X}}\right)$ as a feature, to indicate the entry's ambiguity in the training data.[7] When an X is missing from the table, we fall back on a set of desegmentation rules (El Kholy and Habash, 2012a) and this feature is set to 0. This feature is always 0 for English-Finnish.

Contiguity One advantage of our approach is that it allows discontiguous source words to translate into a single target word. In order to maintain some control over this powerful capability, we create three binary features that indicate the contiguity of a desegmentation. The first feature indicates that the desegmented morphemes were translated from contiguous source words. The second indicates that the source words contained a single discontiguity, as in a word-by-word translation of the "with his blue car" example from Section 5.2.2. The third indicates two or more discontiguities.

Unsegmented LM A 5-gram LM trained on unsegmented target text is used to assess the fluency of the desegmented word sequence.

5.4 Experimental Setup

We train our English-to-Arabic system using 1.49 million sentence pairs drawn from the NIST 2012 training set, excluding the UN data. This training set contains about 40 million Arabic tokens before segmentation, and 47 million after segmentation. We tune on the NIST 2004 evaluation set (1353 sentences) and evaluate on NIST 2005 (1056 sentences). As these evaluation sets are intended for Arabic-to-English translation, we select the first English reference to use as our source text.

Our English-to-Finnish system is trained on the same Europarl corpus as Luong et al. (2010) and Clifton and Sarkar (2011), which has roughly one

[6] Allowing the re-ranker to choose between multiple Ys is a natural avenue for future work.

[7] We also experimented on $\log p(X|Y)$ as an additional feature, but observed no improvement in translation quality.

million sentence pairs. We also use their development and test sets (2000 sentences each).

5.4.1 Segmentation

For Arabic, morphological segmentation is performed by MADA 3.2 (Habash et al., 2009), using the Penn Arabic Treebank (PATB) segmentation scheme as recommended by El Kholy and Habash (2012a). For both segmented and unsegmented Arabic, we further normalize the script by converting different forms of Alif ا آ أ إ and Ya ى ي to bare Alif ا and dotless Ya ى. To generate the desegmentation table, we analyze the segmentations from the Arabic side of the parallel training data to collect mappings from morpheme sequences to surface forms.

For Finnish, we adopt the Unsup L-match segmentation technique of Clifton and Sarkar (2011), which uses Morfessor (Creutz and Lagus, 2005) to analyze the 5,000 most frequent Finnish words. The analysis is then applied to the Finnish side of the parallel text, and a list of segmented suffixes is collected. To improve coverage, words are further segmented according to their longest matching suffix from the list. As Morfessor does not perform any orthographic normalizations, it can be desegmented with simple concatenation.

5.4.2 Systems

We align the parallel data with GIZA++ (Och et al., 2003) and decode using Moses (Koehn et al., 2007). The decoder's log-linear model includes a standard feature set. Four translation model features encode phrase translation probabilities and lexical scores in both directions. Seven distortion features encode a standard distortion penalty as well as a bidirectional lexicalized reordering model. A KN-smoothed 5-gram language model is trained on the target side of the parallel data with SRILM (Stolcke, 2002). Finally, we include word and phrase penalties. The decoder uses the default parameters for English-to-Arabic, except that the maximum phrase length is set to 8. For English-to-Finnish, we follow Clifton and Sarkar (2011) in setting the hypothesis stack size to 100, distortion limit to 6 and maximum phrase length to

20.

The decoder's log-linear model is tuned with MERT (Och, 2003). Re-ranking models are tuned using a batch variant of hope-fear MIRA (Chiang et al., 2008; Cherry and Foster, 2012), using the n-best variant for n-best desegmentation, and the lattice variant for lattice desegmentation. MIRA was selected over MERT because we have an in-house implementation that can tune on lattices very quickly. During development, we confirmed that MERT and MIRA perform similarly, as is expected with fewer than 20 features. Both the decoder's log-linear model and the re-ranking models are trained on the same development set. Historically, we have not seen improvements from using different tuning sets for decoding and re-ranking. Lattices are pruned to a density of 50 edges per word before re-ranking.

We test four different systems. Our first baseline is **Unsegmented**, where we train on unsegmented target text, requiring no desegmentation step. Our second baseline is **1-best Deseg**, where we train on segmented target text and desegment the decoder's 1-best output. Starting from the system that produced 1-best Deseg, we then output either 1000-best lists or lattices to create our two experimental systems. The **1000-best Deseg** system desegments, augments and re-ranks the decoder's 1000-best list, while **Lattice Deseg** does the same in the lattice. We augment n-best lists and lattices using the features described in Section 5.3.4.[8]

We evaluate our system using BLEU (Papineni et al., 2002) and TER (Snover et al., 2006). Following Clark et al. (2011), we report average scores over five random tuning replications to account for optimizer instability. For the baselines, this means 5 runs of decoder tuning. For the desegmenting re-rankers, this means 5 runs of re-ranker tuning, each working on n-best lists or lattices produced by the same (representative) decoder weights. We measure statistical significance using MultEval (Clark et al., 2011), which implements a stratified approximate randomization test to account for multiple tuning replications.

[8]Development experiments on a small-data English-to-Arabic scenario indicated that the Desegmentation Score was not particularly useful, so we exclude it from the main comparison, but include it in the ablation experiments.

5.5 Results

Tables 5.1 and 5.2 report results averaged over 5 tuning replications on English-to-Arabic and English-to-Finnish, respectively. In all scenarios, both 1000-best Deseg and Lattice Deseg significantly outperform the 1-best Deseg baseline ($p < 0.01$).

For English-to-Arabic, 1-best desegmentation results in a 0.7 BLEU point improvement over training on unsegmented Arabic. Moving to lattice desegmentation more than doubles that improvement, resulting in a BLEU score of 34.4 and an improvement of 1.0 BLEU point over 1-best desegmentation. 1000-best desegmentation also works well, resulting in a 0.6 BLEU point improvement over 1-best. Lattice desegmentation is significantly better ($p < 0.01$) than 1000-best desegmentation.

For English-to-Finnish, the Unsup L-match segmentation with 1-best desegmentation does not improve over the unsegmented baseline. The segmentation may be addressing issues with model sparsity, but it is also introducing errors that would have been impossible had words been left unsegmented. In fact, even with our lattice desegmenter providing a boost, we are unable to see a significant improvement over the unsegmented model. As we attempted to replicate the approach of Clifton and Sarkar (2011) exactly by working with their segmented data, this difference is likely due to changes in Moses since the publication of their result. Nonetheless, the 1000-best and lattice desegmenters both produce significant improvements over the 1-best desegmentation baseline, with Lattice Deseg achieving a 1-point improvement in TER. These results match the established state-of-the-art on this data set, but also indicate that there is still room for improvement in identifying the best segmentation strategy for English-to-Finnish translation.

We also tried a similar Morfessor-based segmentation for Arabic, which has an unsegmented test set BLEU of 32.7. As in Finnish, the 1-best desegmentation using Morfessor did not surpass the unsegmented baseline, producing a test BLEU of only 31.4 (not shown in Table 5.1). Lattice desegmentation was able to boost this to 32.9, slightly above 1-best desegmentation, but well

Model	Dev	Test	
	BLEU	BLEU	TER
Unsegmented	24.4	32.7	49.4
1-best Deseg	24.4	33.4	48.6
1000-best Deseg	25.0	34.0	48.0
Lattice Deseg	25.2	34.4	47.7

Table 5.1: Results for English-to-Arabic translation using MADA's PATB segmentation.

Model	Dev	Test	
	BLEU	BLEU	TER
Unsegmented	15.4	15.1	70.8
1-best Deseg	15.3	14.8	71.9
1000-best Deseg	15.4	15.1	71.5
Lattice Deseg	15.5	15.1	70.9

Table 5.2: Results for English-to-Finnish translation using unsupervised segmentation.

below our best MADA desegmentation result of 34.4. There appears to be a large advantage to using MADA's supervised segmentation in this scenario.

5.5.1 Ablation

We conducted an ablation experiment on English-to-Arabic to measure the impact of the various features described in Section 5.3.4. Table 5.3 compares different combinations of features using lattice desegmentation. The unsegmented LM alone yields a 0.4 point improvement over the 1-best desegmentation score. Adding contiguity indicators on top of the unsegmented LM results in another 0.6 point improvement. As anticipated, the tuner assigns negative weights to discontiguous cases, encouraging the re-ranker to select a safer translation path when possible. Judging from the output on the NIST 2005 test set, the system uses these discontiguous desegmentations very rarely: only 5% of desegmented tokens align to discontiguous source phrases. Adding the desegmentation score to these two feature groups does not improve performance, confirming the results we observed during development. The desegmentation score would likely be useful in a scenario where we pro-

Features	dev	test
1-best Deseg	24.5	33.4
+ Unsegmented LM	24.9	33.8
+ Contiguity	25.2	34.4
+ Desegmentation Score	25.2	34.3

Table 5.3: The effect of feature ablation on BLEU score for English-to-Arabic translation with lattice desegmentation.

vide multiple desegmentation options to the re-ranker; for now, it indicates only the ambiguity of a fixed choice, and is likely redundant with information provided by the language model.

5.5.2 Error Analysis

In order to better understand the source of our improvements in the English-to-Arabic scenario, we conducted an extensive manual analysis of the differences between 1-best and lattice desegmentation on our test set. We compared the output of the two systems using the Unix tool *wdiff*, which transforms a solution to the longest-common-subsequence problem into a sequence of multi-word insertions and deletions (Hunt and McIlroy, 1976). We considered adjacent insertion-deletion pairs to be (potentially phrasal) substitutions, and collected them into a file, omitting any unpaired insertions or deletions. We then sampled 650 cases where the two sides of the substitution were deemed to be related, and divided these cases into categories based on how the lattice desegmentation differs from the one-best desegmentation. We consider a phrase to be correct only if it can be found in the reference.

Table 5.4 breaks down per-phrase accuracy according to four manually-assigned categories: (1) **clitical** – the two systems agree on a stem, but at least one clitic, often a prefix denoting a preposition or determiner, was dropped, added or replaced; (2) **lexical** – a word was changed to a morphologically unrelated word with a similar meaning; (3) **inflectional** – the words have the same stem, but different inflection due to a change in gender, number or verb tense; (4) **part-of-speech** – the two systems agree on the lemma, but have selected different parts of speech.

	Lattice Correct	1-best Correct	Both Incorrect
Clitical	157	71	79
Lexical	61	39	80
Inflectional	37	32	47
Part-of-speech	19	17	11

Table 5.4: Error analysis for English-to-Arabic translation based on 650 sampled instances.

For each case covering a single phrasal difference, we compare the phrases from each system to the reference. We report the number of instances where each system matched the reference, as well as cases where they were both incorrect. The majority of differences correspond to clitics, whose correction appears to be a major source of the improvements obtained by lattice desegmentation. This category is challenging for the decoder because English prepositions tend to correspond to multiple possible forms when translated into Arabic. It also includes the frequent cases involving the nominal determiner prefix *Al* "the" (left unsegmented by the PATB scheme), and the sentence-initial conjunction *w+* "and". The second most common category is lexical, where the unsegmented LM has drastically altered the choice of translation. The remaining categories show no major advantage for either system.

5.6 Summary

In this chapter, we have explored deeper integration of morphological desegmentation into the statistical machine translation pipeline. We have presented a novel, finite-state-inspired approach to lattice desegmentation, which allows the system to account for a desegmented view of many possible translations, without any modification to the decoder or any restrictions on phrase extraction. When applied to English-to-Arabic translation, lattice desegmentation results in a 1.0 BLEU point improvement over one-best desegmentation, and a 1.7 BLEU point improvement over unsegmented translation. We have also applied our approach to English-to-Finnish translation, and although segmentation in general does not currently help, we are able to show significant im-

provements over a 1-best desegmentation baseline.

In the next chapter, we explore different methods for translating into unsegmented Arabic while benefiting from segmentation. We experiment with different options for integrating desegmentation in the SMT pipeline such as desegmenting the phrase table or the alignment generated by GIZA++.

Chapter 6

What Matters Most in Morphologically Segmented SMT Models?

In the previous chapter, we introduced one form of integrating desegmentation in the SMT process by desegmenting the decoder's search space. Our lattice desegmentation algorithm resulted in a significant improvement in the translation quality attributed to using features from both segmented and desegmented views of the search space. This brings us to several questions:

- Is it possible to desegment a different component of the SMT pipeline and achieve similar results to the lattice approach?
- Since segmentation improves translation, what steps and components of a phrase-based statistical machine translation pipeline benefit the most from segmenting the target language?

In this chapter, we explore different options of integrating desegmentation with the SMT processes. We test several scenarios that differ primarily in when desegmentation is applied, showing that the most important criterion for success in segmentation is to allow the system to build target words from morphemes that span phrase boundaries. We also investigate the impact of segmented and unsegmented target language models (LMs) on translation quality. We show that an unsegmented LM is helpful according to BLEU score, but also leads to a drop in the overall usage of compositional morphology, bringing it to well below the amount observed in human references.

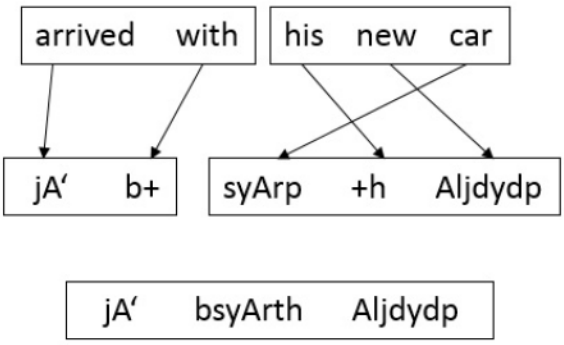


Figure 6.1: An illustration of one-to-one correspondence between Arabic morphemes and English words. Arabic text is segmented using the PATB tokenization scheme, and shown in Buckwalter transliteration.

6.1 Introduction

Segmentation has repeatedly been shown to improve translation into or out of morphologically complex languages by splitting relevant morphological affixes into independent tokens. Segmentation as a pre-processing step brings several benefits to translation:

- **Correspondence** with morphologically simple languages, such as English is improved. In Figure 6.1, segmenting *bsyArth* allows one-to-one links for "with", "his" and "car".

- By building models over morphemes, rather than words, **data sparsity** is reduced.

- By allowing morphemes with clear syntactic roles to be translated independently, we increase our **expressive power** by creating new lexical translations. For example, using the two phrase-pairs in Figure 6.1 results in a new word after desegmentation (b+ syArp +h $\Rightarrow$ bsyArth), which might not have existed in the training data.

However, there is also a price to be paid. While morpheme-level models are more resistant to data sparsity, they account for less context than word-level models, make stronger independence assumptions, and they are less efficient statistically, in that they devote probability mass to sequences containing ille-

gal words. Furthermore, when segmentation is applied to the target language, the process must be reversed at the end of the pipeline to present the output in a readable format. This **desegmentation** step complicates our pipeline, and can introduce errors.

Our work in this chapter is inspired by two recent contributions that attempt to combine the advantages of word- and morpheme-based models. Luong et al. (2010) combine word and morpheme views in a desegmented phrase table, allowing morphemes to reduce sparsity while words expand context, and eliminating the need for a separate desegmentation step. Their word-boundary-aware morpheme-level phrase extraction technique restricts phrase boundaries so that no target phrase can begin with a suffix or end with a prefix. This allows them to desegment each target phrase independently, enabling the use of both word- and morpheme-level language models during decoding. However, this phrase-table desegmentation approach lacks the expressive power that comes from translating morphemes independently.

In the previous chapter, we introduced the lattice desegmentation approach, which comes close to combining all the advantages of word and morpheme views. By desegmenting a lattice that compactly represents many translation options, and rescoring it with a word-level language model, we avoid restricting the phrase table. However, by delaying desegmentation until rescoring, the approach loses Luong et al. (2010)'s advantage of full decoder integration.

In this chapter, we present an experimental study of English-to-Arabic translation that is designed to better understand the impact of various trade-offs when translating into a morphologically segmented target language, and to identify what aspects of segmentation are most beneficial to translation. The benefits of segmentation can impact several components in the SMT pipeline: the alignment model, the translation table, and the various language and translation models. Throughout this study, we investigate the effect of varying the point in the SMT pipeline where the segmentation is reversed. In addition, we attempt to combine word- and morpheme-level models within the decoder as much as possible.

Our experimental study provides three novel insights. First, we present strong evidence indicating that the ability to build target words across phrase boundaries is the most important property of target language segmentation. This implies that phrase table desegmentation, the only published desegmentation technique that has been fully integrated into decoding, gives up segmentation's primary advantage. Second, we draw a previously unobserved connection between the use of an unsegmented LM and the decoder's overall use of compositional morphology; we show that although unsegmented LMs tend to increase BLEU score, they also reduce the system's use of morphological affixes to well below that of a human. Finally, we present the first direct comparison between phrase table desegmentation (Luong et al., 2010) and lattice desegmentation (Salameh et al., 2014).

6.2 Background

Our work builds on earlier studies of automatic morphological segmentation and its impact on SMT. There are many ways to segment syntactically relevant affixes from stems. Supervised techniques may either pass through an intermediate morphological analysis (Habash et al., 2009), or directly segment the character stream (Green and DeNero, 2012); recent work on supervised Arabic segmentation focuses primarily on adaptation to dialects (Habash et al., 2013; Monroe et al., 2014). There are also a host of unsupervised techniques (Creutz and Lagus, 2005; Lee et al., 2011; Sirts and Goldwater, 2013), which provide valuable language portability, but which generally fall behind supervised methods when labeled data is available.

There is a large body of work studying the best form of segmentation when translating from a morphologically complex source language (Sadat and Habash, 2006; Stallard et al., 2012), where the segmentation can be used as a simple preprocessing step, or to create an input lattice (Dyer et al., 2008). Recently, there has been a growing interest in segmentation on the target side (Oflazer and Durgar El-Kahlout, 2007), which introduces a question of how to perform proper desegmentation (Badr et al., 2008). El Kholy and

Habash (2012a) have conducted a thorough exploration of the various segmentation and desegmentation options for English to Arabic translation, and we follow their work when designing our test bed.

6.3 Methods

When translating into a segmented target language, such as Arabic, the segmentation will need to eventually be reversed for the output to be readable. The key insight driving our experiments is that by varying the point in the SMT pipeline where this reversal occurs, we can alter which models are based on morphemes and which are based on words, and thereby determine which components most benefit from segmentation. We assume a phrase-based SMT architecture similar to that of Moses (Koehn et al., 2007), but most of our observations hold for hierarchical and tree-based models. In all of our approaches, we desegment using a mapping table that counts the segmentations performed on the target side of our training data. The table uses counts of word-segmentation pairs to map each morpheme sequence back to its most likely unsegmented word form. We back off to manually crafted rules in cases where the segmented form does not exist in the mapping table (El Kholy and Habash, 2012a).

Method	Unsegmented	Alignment Deseg.	Phrase Table Deseg.	One-best Deseg.	Lattice Deseg.
Desegment before:	Never segment	Phrase extraction	Decoding	Evaluation	Evaluation
Alignment model	Word	Morph	Morph	Morph	Morph
Lexical weights	Word	Word	Morph	Morph	Morph
Language model	Word	Word	Word	Morph	Morph + Word
Tuning	Word	Word	Word	Morph	Morph then Word
Flexible boundaries?	No	No	No	Yes	Yes

Table 6.1: Desegmentation scenarios and their effect on the components of a typical SMT system.

Table 6.1 summarizes the effect of the desegmentation point on the components of a typical SMT system, indicating which components are built using morphemes and which are built using words. Most components should be familiar, but the last row introduces **flexible boundaries**, a concept that will be central to our study. This property of the phrase table indicates whether phrases can have unattached affixes at their left or right boundaries. Systems without flexible boundaries cannot combine morphemes across phrases to create translations that were not already seen in the parallel text; as such, this property has a large impact on a system's expressive power.

We describe our comparison systems in turn, each corresponding to a column in Table 6.1. We also describe a segmented language model feature, which can be added to any system that uses a word-level phrase table.

6.3.1 Baselines

We rely on two main baselines to evaluate what matters most in segmented models. An **unsegmented** system leaves the Arabic target unsegmented and uses an unsegmented language model. This model suffers from data sparsity and poor English-Arabic word correspondence. The decoder always outputs morphologically correct Arabic words, as it does not require a desegmentation step.

Meanwhile, **one-best desegmentation** segments the Arabic target language before training begins, and the decoder's output is generated in segmented form. As a post-processing step, the one-best output is desegmented using a mapping table and desegmentation rules. All of the component models used during decoding are based on morphemes instead of words. The segmented models are intended to help alleviate data sparsity and improve token correspondence. Unlike the unsegmented system, this system requires a desegmentation step, which can produce morphologically incorrect words.

6.3.2 Alignment Desegmentation

Our unsupervised alignment models (Brown et al., 1993; Och and Ney, 2003) are sensitive both to poor word-to-word correspondence and to data sparsity

issues. They are also at the very start of the SMT pipeline; they impact nearly all other downstream models. Therefore, it would be reasonable to suspect that the primary benefit of segmentation could come from improved word alignment. Alignment desegmentation allows us to test this theory by desegmenting immediately after alignment.

More specifically, we segment the target side as pre-processing. After word alignment, we replace the segmented Arabic training data with its unsegmented form. Note that this desegmentation is perfect, as we can always refer to the original sentence to resolve any ambiguities. This is accompanied by desegmenting alignment links by replacing each morpheme index with the index of the unsegmented word that now contains the morpheme. As one would expect, this leads to an increase in the number of one-to-many alignments. Training is then resumed with these links and the unsegmented target. Other than having its alignment model benefit from segmentation, this system has the same properties of an unsegmented system: all remaining component models are based on words. Since all morphemes are desegmented well before decoding begins, it clearly cannot use flexible boundaries to build new words.

6.3.3 Phrase Table Desegmentation

Our next desegmentation point is after phrase extraction, resulting in a system where we segment the text, align the morphemes, perform phrase extraction over morphemes, and then desegment the resulting tables. Following Luong et al. (2010), we first remove all phrases that have target sides with flexible boundaries, which allows us to desegment each remaining target phrase independently. The result is a desegmented phase table. Note that we leave the various scores associated with each phrase-pair unchanged.

This model is similar to alignment desegmentation described in the previous section in that all remaining components and operations are based on words. However, there are two key differences. First, the lexical weights of each phrase are calculated over morphemes rather than words. Second, the phrase-length limit is applied at the morpheme level rather than at the word level. We use this scenario to test the utility of morpheme-level lexical weights.

This system is related to, but not identical to the work of Luong et al. (2010). Their system actually merges tables from an unsegmented model with those from phrase table desegmentation; they investigate a number of methods to combine the scores across tables. In addition, they incorporate both segmented and unsegmented language models, which is a difference that we address in the next section.

6.3.4 Segmented LM Scoring in Desegmented Models

Both alignment desegmentation and phrase table desegmentation rely on an unsegmented language model, as they naturally decode directly into a desegmented target language. We experiment with augmenting both of these systems with an extra feature: a segmented language model. For each Arabic target word, we add its segmented form to the phrase table as an extra factor (Koehn and Hoang, 2007). We insert this factor after phrase extraction, so it has no impact on alignment or the calculation of translation model scores. The factor merely gives us access to the segmented morphemes during decoding. The decoder uses this factor to apply a segmented language model during each hypothesis extension.

Although the segmented language model spans a shorter context, its scores benefit from the reduced data sparsity that comes from modeling morphemes. In particular, it can unveil whether attaching two hypotheses is grammatical. For example, the unsegmented language model score for the consecutive target phrases كل مشاكلنا *kl mšAklnA* "all our problems" وخلافاتنا *wxlAfAtnA* "and conflicts" is relatively low. Scoring their segmented representation [kl mšAkl +nA] [w+ xlAfAt +nA] leads to a more optimistic score, as the segmented language model assesses the morpheme sequence using 4-grams and trigrams, while the unsegmented model scores the word sequence with unigrams and bigrams.

6.3.5 Lattice Desegmentation

We compare the previous approaches to the lattice desegmentation approach (Chapter 5) and place it in Table 6.1 for reference. A system built entirely

over morphemes outputs a pruned lattice that compactly represents its hypothesis space. This lattice is then desegmented by composing it with a finite state transducer that maps morpheme sequences into words. By rescoring the desegmented lattice with new features, the system benefits from having both a segmented and desegmented view of the search space. The added features include discontiguity features, as well as an unsegmented language model. The discontiguity features indicate whether a desegmented word came from one contiguous morpheme sequence, two discontiguous sequences, or more.

6.4 Experimental Setup

We train our English-to-Arabic system using 1.49 million sentence pairs drawn from the NIST 2012 training set, excluding the UN data. This training set contains about 40 million Arabic tokens before segmentation, and 47 million after segmentation. We tune on the NIST 2004 evaluation set (1353 sentences) and evaluate on NIST 2005 (1056 sentences). We also report a second test, which tunes on the NIST 2006 evaluation set (1664 sentences) and evaluates on NIST 2008 (1360 sentences) and 2009 (1313 sentences). NIST 2004 and 2005 datasets have sentences from newswire, while NIST 2006/2008/2009 have sentences drawn from newswire and the web. These evaluation sets are intended for Arabic-to-English translation, and therefore have multiple English references. As we are translating into Arabic, we select the first English reference to use as our source text, and use the Arabic source as our single reference translation.

6.4.1 Segmentation

For Arabic, morphological segmentation is performed by MADA 3.2 (Habash et al., 2009), using the Penn Arabic Treebank (PATB) segmentation scheme as recommended by El Kholy and Habash (2012a). For both segmented and unsegmented Arabic, we further normalize the script by converting different forms of Alif and Ya to bare Alif and dotless Ya. In order to generate the desegmentation table, we analyze the MADA segmentations from the Arabic

Model	mt05	mt08	mt09
Unsegmented	32.8	15.0	19.0
Alignment Deseg.	33.4	15.4	19.1
with Segmented LM	33.7	15.4	19.4
Phrase Table Deseg.	33.4	15.5	19.3
with Segmented LM	33.6	15.6	19.7
1-best Deseg.	33.7	15.7	20.2
without flexible boundaries	32.9	15.4	19.4
Lattice Deseg.	34.3	16.4	20.5

Table 6.2: BLEU scores on each of the methods described in section 6.3 . MT05 results are tuned using NIST MT04. Results on NIST MT08 and MT09 datasets are tuned on MT06 dataset.

side of the parallel training data to collect mappings from morpheme sequences to surface forms.

6.4.2 Systems

We align the parallel data with GIZA++ (Och et al., 2003) and decode using Moses (Koehn et al., 2007). The decoder's log-linear model includes a standard feature set. Four translation model features encode phrase translation probabilities and lexical weights in both directions. Seven distortion features encode a standard distortion penalty as well as a bidirectional lexicalized reordering model. A KN-smoothed 5-gram language model is trained on the target side of the parallel data with SRILM (Stolcke, 2002). Finally, we include word and phrase penalties. The decoder uses Moses' default search parameters, except that the maximum phrase length is set to 8. The decoder's log-linear model is tuned with MERT (Och, 2003). Following Salameh et al. (2014), the tuning of the re-ranking models for lattice desegmentation is performed using a lattice variant of hope-fear MIRA (Cherry and Foster, 2012); lattices are pruned to a density of 50 edges per word before re-ranking. We evaluate our system using BLEU (Papineni et al., 2002).

6.5 Results

Table 6.2 shows the results of our translation quality experiments. In previous sections, we mentioned several factors that might contribute to the quality improvements found with segmented models. Beyond the raw ranking of systems, we can use the commonalities and differences between these systems to draw some broad conclusions of what aspects of a segmented system are most important.

6.5.1 Decoder Integration

Lattice Desegmentation performs best overall, which is not entirely surprising, as it has access to all of the information present in the other systems. Notably, it outperforms Phrase Table Desegmentation; this is the first time to our knowledge that the two have been compared directly.

The main disadvantage of Lattice Deseg, which is not present in Alignment and Phrase Table Deseg, is the lack of decoder integration of its unsegmented view of the target; instead, it is handled by re-ranking a lattice in post-processing. In fact, the top two systems, Lattice Deseg and 1-Best Deseg, are also the only two systems without access to unsegmented information in the decoder. This suggests that the benefits of decoder integration are not sufficient to overcome the trade-offs currently demanded by integration.

6.5.2 Flexible Boundaries

What is perhaps more surprising is that neither Alignment Deseg nor Phrase Table Deseg are able to match the 1-best Deseg scenario. With the benefit of added segmented language models, both of these systems have access to almost all 1-best Deseg's information and more, yet they fail to match its translation quality in every test. What both systems lack with respect to 1-best Deseg is flexible phrase boundaries, which allow the creation of new translations across phrases. To confirm the importance of flexible boundaries, we created a new version of 1-best Deseg by pruning all phrases with flexible boundaries from the phrase table, and then re-tuning. The resulting system loses 0.6 BLEU on

average, which is more than half of the 0.9 difference between Unsegmented and 1-best Deseg. We conclude that flexible boundaries are one of the most important aspects of a segmentation scenario.

6.5.3 Language Models

Both Align Deseg and Phrase Table Deseg show consistent, albeit small, improvements from the addition of a segmented LM. In order to assess the importance of the unsegmented LM, we consider 1-best Deseg without flexible boundaries, and Phrase Table Deseg with Segmented LM. These two systems have exactly the same output space, as their respective phrase tables are constructed from morpheme-level phrase extraction followed by pruning flexible boundaries. Furthermore, both systems use a segmented LM and lexical weights built over morphemes. Their only differences are that Phrase Table Deseg uses an unsegmented LM and unsegmented tuning, resulting in BLEU scores that are higher by 0.4 on average. Similarly, a unsegmented LM is one of the main differences between Lattice Deseg and 1-best Deseg, with the others being unsegmented tuning and discontiguity features. Although we have not isolated the unsegmented LM perfectly, these results indicate that it is valuable.

6.5.4 Lexical Weights

The primary difference between Alignment Deseg and Phrase Table Deseg is that the latter uses morpheme-level lexical weights.[1] Without a segmented LM, we see a 0.1 average BLEU advantage for Phrase Table Deseg, increasing to 0.2 when a segmented LM is included. Unfortunately, these improvements are not consistent across test sets. This suggests that there may be an advantage from morpheme-based lexical weights, but it is certainly not large.

Model	mt05	mt08	mt09
Reference	15.9%	18.1%	18.9%
Unsegmented	12.0%	12.2%	12.6%
Alignment Deseg.	11.6%	11.0%	11.8%
with Segmented LM	11.7%	11.2%	12.0%
Phrase Table Deseg.	11.3%	10.1%	11.2%
with Segmented LM	11.6%	10.5%	11.4%
1-best Deseg.	16.1%	18.2%	19.2%
without flexible boundaries	14.2%	14.7%	15.4%
Lattice Deseg.	10.0%	11.5%	12.2%

Table 6.3: Percentage of words in the SMT output that have non-identity morphological segmentations.

6.6 Analysis

Our translation quality comparison indicates that flexible boundaries are the most important property of a target segmentation scenario, so we examined them in greater detail. Phrase pairs with flexible boundaries account for roughly 12% of phrases used in the final output of our 1-Best Deseg system.

We performed a detailed analysis to see if the flexible boundaries were used to produce novel words; that is, words that were not seen in the target side of the training data. Roughly 3% of the desegmented types generated by the 1-best-desegmentation system are novel. We randomly selected 40 novel words from each test set to analyze manually. First, none of these desegmented words appear in the reference, and therefore, they have no positive impact on BLEU. Furthermore, 64 of the 120 selected words violate the morphological rules of Arabic. Looking instead at the novel words in the reference, only 115 reference words could not be found in the Arabic side of our training data. Of these, only 37 could be constructed from morphemes found in our training set. This means that there is only a small number of opportunities to better match the reference by producing a novel word. Together, these two pieces of analysis strongly suggest that the advantage of flexible boundaries comes from creating new translation options for a given source sequence, rather than from creating

[1]The other difference is the calculation of the phrase length limit, which favors Alignment Deseg, as its word-based limit allows more phrases overall.

novel words.

We were able to compute statistics on flexible boundaries for only two of our systems, because the other three disallow them entirely. In order to characterize all five systems, along with the human references, we measured overall affix usage by counting decomposable words. Table 6.3 shows the percentage of words in the Arabic translations that have non-identity morphological segmentations when processed by MADA. In terms of affix usage, the 1-best Deseg method tracks the Reference very closely, while all remaining scenarios show a substantial drop in usage of decomposable words. Most surprisingly, Lattice Deseg is included in this group, even though its BLEU scores are higher than 1-best Deseg. Since 1-Best Deseg's most prominent characteristic is its lack of an unsegmented LM, this suggests that unsegmented LMs may dramatically impact affix usage. Note that flexible boundaries do not (fully) account for the gap in affix usage, as the 1-best Deseg still has noticeably higher usage of decomposable words, even with flexible boundaries removed. This implies that Lattice Deseg and the various fully integrated desegmentations could be improved by attempting to directly manipulate their usage of decomposable words, perhaps through a specialized feature.

As a final piece of analysis, we also investigated the impact of different n-gram orders for segmented LMs. Most of the scenarios proposed here add an unsegmented LM to a segmented system, and the most obvious advantage of an unsegmented LM is that it accounts for more context than a segmented LM. However, this only holds if we force both LMs to have the same n-gram order. To see if higher order segmented LMs would improve translation, we experimented with different n-gram orders for our 1-best Deseg system. As we increased the segmented n-gram order from 5 to 8, we saw no improvement over the 5-gram LM used throughout this chapter. In fact, BLEU score began to drop after $n = 6$. This suggests that the advantage of adding an unsegmented LM cannot be emulated by increasing the order of the segmented LM.

6.7 Summary

In this chapter, We have presented an experimental study on translation into segmented target languages by creating models that apply desegmentation at different points in the translation pipeline. We have provided evidence that access to phrases with flexible boundaries is a crucial property for a successful segmentation approach. We have also examined the impact of unsegmented LMs, showing that although they are helpful according to BLEU, they also hinder the generation of morphologically-complex words. This suggests that current methods could be improved by attempting to increase their use of morphological affixes.

Chapter 7

Integrating Morphological Desegmentation into Phrase-based Decoding

In the last two chapters, we attempted to benefit from desegmentation through integrating it with various SMT components. We also confirmed that the best integration approach is to desegment the search space encoded as a lattice. Our analysis showed that the use of morpheme-level and word-level features improves the quality of translation and validates the correctness of the desegmented forms.

In all of the previous approaches, we had to interrupt the SMT pipeline at some point to apply desegmentation. In this chapter, we attempt to integrate desegmentation directly into decoding, such that morphemes that contribute to forming a word are desegmented on the fly. This allows word-level features to be extracted from the desegmented forms and to be considered throughout the entire search space. We elaborate on the challenges that arise due to this integration. Our results on a large-scale, English to Arabic translation task show significant improvement over the 1-best desegmentation baseline.

7.1 Introduction

Morphological segmentation is typically performed as a pre-processing step before the training phase, which results in a model that translates the source language into segmented target language. *Desegmentation* is the process of

transforming the segmented output into a readable word sequence, which can be performed using a table lookup combined with a small set of rules. Desegmentation is usually applied to the 1-best output of the decoder. However, this pipeline suffers from error propagation: errors made during decoding cannot be corrected, even when desegmentation results in an illegal or extremely unlikely word. Two principal types of solutions that have been proposed for this problem are *rescoring* and *phrase-table desegmentation*.

The rescoring approach desegments either an n-best list (Oflazer and Durgar El-Kahlout, 2007) or lattice (Chapter 5) , and then re-ranks with features that consider the desegmented word sequence of each hypothesis. Rescoring features include the score from an unsegmented target language model and contiguity indicators that flag the target words constructed from contiguous source tokens. Rescoring widens the desegmentation pipeline, thus allowing desegmentation features to reduce the number of translation errors. However, these features are calculated only for a subset of the search space, and the extra rescoring step complicates the training and translation processes.

Phrase-table desegmentation (Luong et al., 2010) also translates into a segmented target language, but alters training to perform word-boundary-aware phrase extraction. The extracted phrases are constrained to contain only complete target words, without any *dangling affixes* (phrases with flexible boundaries). With this restriction in place, the phrase table can be desegmented before decoding begins, allowing the decoder to track features over both the segmented and desegmented target. This ensures that desegmentation features are integrated into the complete search space, and side-steps the complications of rescoring. However, we show experimentally in Chapter 6 that these benefits are not worth giving up the phrase-pairs eliminated by word-boundary-aware phrase extraction.

We present a method for decoder-integrated desegmentation that combines the strengths of these two approaches. Like a rescoring approach, it places no restrictions on what morpheme sequences can appear in the target side of a phrase pair. Like phrase-table desegmentation, its desegmentation features are integrated directly into decoding and considered throughout the entire

search space. We achieve this by augmenting the decoder to desegment hypotheses on the fly, allowing the inclusion of an unsegmented language model and other features. Our results on a large-scale, NIST-data English to Arabic translation task show significant improvements over the 1-best desegmentation baseline, and match the performance of the state-of-the-art lattice desegmentation approach (Chapter 5) , while eliminating the complication and cost of its rescoring step. Our approach is implemented as a single stateful feature function in Moses (Koehn et al., 2007).

7.2 Method

Our approach extends the multi-stack phrase-based decoding paradigm to enable the extraction of word-level features inside morpheme-segmented models.[1] We assume that the target side of the parallel corpus has been segmented into morphemes with prefixes and suffixes marked. This allows us to define a *complete word* as a maximal morpheme sequence consisting of 0 or more prefixes, followed by at most one stem, and then 0 or more suffixes. We also assume access to a desegmentation function that takes as input a morpheme sequence matching the above definition, and returns the corresponding word as output. For our Arabic experiments, we use a table-based desegmentation scheme that falls back on a small set of rules for sequences not found in its table. The output of a phrase-based decoder is built from left to right, and at each step, a hypothesis is expanded with a phrasal translation of a previously uncovered source segment. *In-decoder desegmentation* monitors the target sequence of each translation hypothesis as it grows, detecting morpheme sequences that correspond to complete words and desegmenting them on the fly to generate new features.

The task of determining whether a word is complete is non-trivial. We are never sure if we will see another suffix as we expand the hypothesis, so we can only recognize a complete word as we begin the next word. For example, take

[1]The ideas presented here could also be applied to hierarchical decoding, which would require generalizing them to account for right context as well as left.

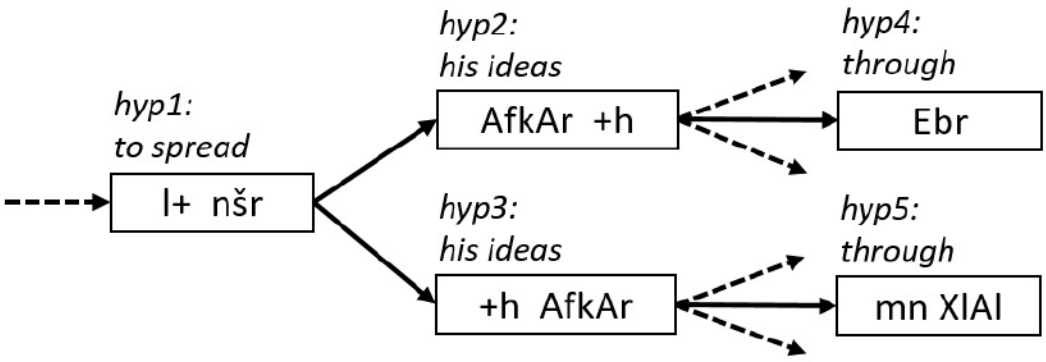


Figure 7.1: Decoding the Arabic translation of the phrase *"to spread his ideas through"*.

hyp1 in Figure 7.1. This hypothesis ends with a stem *nšr*, which may end a complete word, as is the case when we expand to hyp2, or may represent a word that is still in progress, which occurs as we extend to hyp3. This means that the word-based scoring of the morpheme sequence *l+ nšr* must be delayed or approximated until we know what follows. A related challenge involves scoring phrase-pairs out of context, as is required for future-cost estimates. Take, for example, the target phrase *+h AfkAr* added by hyp3 in Figure 7.1. Without the context, we have insufficient information at the left boundary to score *+h* with word-based models, while *AfkAr* at the right boundary may or may not form a complete word. Here, there is no choice but to approximate. The quality of these approximations and the length of our delays will determine how effective our new features will be when incorporated into beam search.

7.2.1 Decoder Integration

A typical phrase-based decoder represents a hypothesis with a state that contains the information to guide search and calculate features, such as the source coverage vector and the target context for the language model. Hypotheses with identical states can be recombined to improve search efficiency. We augment the state with two structures: (1) a buffer Q containing all of the morphemes that contribute to the current word in progress, represented as a queue of tokens; and (2) n-gram context C for the word-level target language model. The search's initial state begins with an empty Q and with *n-1*

Algorithm 2 Desegmentation State Update

```
Input: State variables Q, C
Input: Extending phrase P
for each token t in P do
  if t cannot continue the word in Q then
    W = Desegmentation of tokens in Q
    Extract word-level features for W
    (Word-level LM score is p(W|C))
    Update current feature vector
    Update C with W
    Empty Q
  Append token t to Q
```

beginning-of-sentence tokens in C.

When a state is extended with a target phrase P, we update the in-decoder desegmentation structures Q and C with Algorithm 2. Tokens are appended to Q until a token t would begin a new word, at which point the tokens from Q are desegmented and the resulting word is used to calculate features and update the target context. Following the lower decoding path in Figure 7.1, Q would be emptied and desegmented first during hyp3 when $t = AfkAr$, calculating features for $W = lnšrh$.

The main cost of in-decoder desegmentation comes from maintaining the context necessary to evaluate the n-gram, word-level language model. As each desegmented word in C will correspond to at least one segmented token, the system's effective language-model order in terms of segmented tokens will frequently be much larger than n. Storing larger language-model contexts make it less likely that states will be equal to one another, which reduces the amount of recombination the system can do, and increases the number of states that must be expanded during search.

7.2.2 Delayed and Optimistic Scoring

In the above approach, desegmentation and feature scoring are applied only when a complete word is formed. We refer to this as **delayed scoring** because the features for a token are not applied until other tokens have been added to the hypothesis. For example, in Figure 7.1, the tokens *l+ nšr* added in hyp1

are not evaluated with word-level features until hyp2 or hyp3 completes the word. This delay results in inaccurate scoring of hypotheses, as the cost from these tokens is hidden until Q is emptied. These inaccuracies can lead to poor pruning choices and search errors during beam search.

Alternatively, we can perform **optimistic scoring**, which tries to score the contents of Q as early as possible. In this case, we assume that the contents of Q form a complete word, without waiting for the next token to confirm it. With each hypothesis extension, when the last token in P is processed and added to the queue, we desegment the contents of Q and extract features, but without emptying Q. The scores of these features are cached in a variable S that does not affect recombination, as the scores are deterministic given Q, C and the model. When a later token confirms the end of the word, we subtract S from the scores derived from the actual desegmented word, to account for our earlier approximation. Note that for a Q containing only a prefix, we must still delay scoring.

7.2.3 Features

Three features are extracted from each desegmented form. An unsegmented n-gram language model scores W in the context of C, as shown in Algorithm 2. We also implement the contiguity features as in section 5.3.4. These indicators check if the desegmented form is generated from a contiguous block of source tokens, a block with 1 discontiguity, or a block with multiple discontinuities. Finally, most phrase-based decoders incorporate a word penalty that counts the number of target words in a hypothesis. In our scenario, this can count segmented morphemes or desegmented words. We try both in our experiments.

For future cost estimates, we must also provide out-of-context feature scores for each phrase-pair in our system. To do so, we ignore suffixes appearing at the beginning of a target phrase and prefixes appearing at the end. We assume that the remaining tokens form complete words, and desegment and score them to provide out-of-context scores.

7.3 Experiments

We use the NIST 2012 dataset (1.49 million sentence pairs excluding UN pairs) to train an English-to-Arabic system. The system is tuned with the NIST 2004 (1353 pairs) evaluation set and tested using NIST 2005 (1056 sentences) and the newswire portion of NIST 2008 (813 pairs) and NIST 2009 (586 pairs). As there are multiple English reference translations provided for these evaluation sets, we use the first reference as our source text.

The Arabic part of the training set is morphologically segmented by MADA 3.2 (Habash et al., 2009) using the PATB segmentation scheme. Variants of Alif and Ya characters are uniformly normalized. We generate a desegmentation table from the Arabic side of the training data by collecting mappings of segmented forms to surface forms.

We align the parallel data with GIZA++ (Och et al., 2003), and decode with Moses (Koehn et al., 2007). The decoder's log-linear model includes a standard feature set. KN-smoothed 5-gram language models are trained on both the segmented and unsegmented views of the target side of the parallel data. We experiment with word penalties based either on morphemes or desegmented words. The decoder uses Moses' default search parameters, except for the maximum phrase length, which is set to 8, and the translation table limit, which is set to 40 [2]. The decoder's log-linear model is tuned with MERT (Och, 2003) using the unsegmented form of the Arabic development set. We evaluate with BLEU (Papineni et al., 2002), and test statistical significance with multeval (Clark et al., 2011) over 3 random tuning replications .

We test four systems that differ in their desegmentation approach. The **NoSegm.** baseline involves no segmentation. The **One-best** baseline translates into segmented Arabic and desegments the decoder's 1-best output. The **Lattice** system is the lattice-desegmentation approach of Chapter 5. We implement our **in-Decoder** desegmentation approach as a feature functions in

[2]We experimented with different translation options limits on 1-best-deseg system but found no improvement above 40 options

System	WP	mt05	mt08	mt09
NoSegm.	word	33.2	18.6	25.6
One-best	morph.	33.8	19.1	26.8
Lattice	morph.	34.4	**19.7**	**27.4**
Delayed	morph.	34.1	19.4	27.0
	word	34.1	19.5	26.8
Optimistic	morph.	34.2	19.6	27.2
	word	**34.5**	**19.7**	27.2

Table 7.1: Evaluation of the desegmentation methods using BLEU score. Both Delayed and Optimistic refer to in-Decoder Desegmentation method used. WP shows whether Word Penalty feature is based on a complete desegmented word or a morpheme.

Moses, testing scoring variants (delayed vs. optimistic), and word penalty variants (morpheme vs. word).

Table 7.1 shows the results on three NIST test sets, each averaged over 3 tuning replications. The lattice approach is significantly better than the 1-best system, which in turn is significantly better than the unsegmented baseline. Our **Optimistic** in-decoder approach with word penalty calculated on word tokens is significantly ($p < 0.05$) better than the 1-best approach, and effectively matches the accuracy of the more complex lattice approach. Typically, one would hope to surpass a rescoring approach with decoder integration; however, our lattice implementation fully searches its lattice, even if composition with the word-level language model causes the lattice to explode in size. A lattice beam search that dynamically calculates word-level language model scores would provide a more fair, and more efficient, comparison point.

All of the systems with word-level features improve over the 1-best model, as their features penalize desegmentations resulting in illegal words or unlikely word sequences. We see a small, consistent benefit from optimistic scoring. Error analysis reveals that translations with many consecutive stems benefit the most from this variant, which makes sense, as our approximations would be exact in these cases. We were surprised to see differences between word penalty variants, but a penalty based on desegmented word-forms provides the system with greater control over the number of words that appear in its final output, which can be important.

7.4 Summary Results

For the sake of complete comparison, we reran our experiments using the same Moses release (version 2.1.1), same hyper-parameters and same tuning and evaluation datasets as in section 7.3. The main differences between the experiments in this chapter and the ones in previous chapters is in setting the number of translation-table-options to 40 (previously was 20) and using only NIST MT04 for tuning. Also, we removed the web data from NIST MT08 and MT09 and only used the newswire portions. We summarize the results in Table 7.2 with the methods introduced in Chapters 5, 6 and 7.

Ref.	System	WP	mt05	mt08	mt09
Ch. 5	**NoSegm.**	word	33.2	18.6	25.6
Ch. 6	**Alignment Deseg.**	word	33.3	19.2	26.1
Ch. 6	**with Segmented LM**	word	33.0	18.8	25.9
Ch. 6	**Phrase Table Deseg.**	word	33.4	19.1	25.9
Ch. 6	**with Segmented LM**	word	33.7	19.2	26.4
Ch. 5	**One-best**	morph.	33.8	19.1	26.8
Ch. 6	**without flexible boundaries**	morph.	33.0	18.8	26.1
Ch. 5	**1000-best-list**	morph.	34.1	19.3	27.2
Ch. 5	**Lattice**	morph.	34.4	**19.7**	**27.4**
Ch. 7	**Delayed in-Decoder**	morph.	34.1	19.4	27.0
		word	34.1	19.5	26.8
Ch. 7	**Optimistic in-Decoder**	morph.	34.2	19.6	27.2
		word	**34.5**	**19.7**	27.2

Table 7.2: Evaluation of different desegmentation methods presented in this thesis using BLEU score. The *Ref* column mentions the chapter where the desegmentation methods was first introduced.

The results show similar pattern to the ones in previous chapters. The BLEU score of 1000-best-list desegmentation lies between the 1-best-deseg and lattice desegmentation. Alignment and phrase-table deseg. scores lie between the NoSegm and 1-best-desegm scores. Also, one-best without phrases with flexible boundaries achieved scores close to the NoSegm (except on MT09). The table shows that the overall conclusions hold while relative comparisons may have slightly changed on some experiments.

In this chapter, we tackled a challenging problem of integrating deseg-

mentation into decoding, and got scores similar to the lattice desegmentation approach, but with a method that employs a substantially simpler pipeline. At this point, we have seen one form of Arabic morphological transformation through segmentation that resulted with several benefits. In the next chapter, we explore a different transformation on a sentiment analysis task, namely, Arabic lemmatization.

Chapter 8

Sentiment after Translation: A Case-Study on Arabic Social Media Posts

In the previous chapters, we applied segmentation on Arabic to improve translation from English into Arabic. Segmentation is one form of language transformation for handling morphologically complex languages and decreasing token sparsity. In this chapter, we explore morphological analysis (lemmatization) as an alternative approach for morphologically complex languages.

In general, when text is translated from one language into another, sentiment is preserved to varying degrees. In this chapter, we use Arabic social media posts as a stand-in for source language text, and determine loss in sentiment predictability when they are translated into English, manually and automatically. As benchmarks, we use manually and automatically determined sentiment labels of the Arabic texts. We create a state-of-the-art Arabic sentiment analysis system and show it has the best accuracy on lemmatized forms of Arabic sentences. We also improve our system by creating Arabic lexicon that is automatically generated from lemmatized-form of Arabic tweets. We also show that sentiment analysis of English translations of Arabic texts produces competitive results, w.r.t. Arabic sentiment analysis. We discover that even though translation significantly reduces the human ability to recover sentiment, automatic sentiment systems are still able to capture sentiment information from the translations.

8.1 Introduction

Automatic sentiment analysis of text, especially social media posts, has a number of applications in commerce, public health, and public policy development. However, a vast majority of prior research on automatic sentiment analysis has been on English texts. Furthermore, many sentiment resources essential to automatic sentiment analysis (e.g., sentiment lexicons) exist only in English. Thus there is a growing need for effective methods for analyzing text from other languages such as Arabic and Chinese, especially posts on social media. There has also been marked progress in automatic translation of texts, especially from other languages into English. Thus, instead of building source-language specific sentiment analysis systems, one can translate the texts into English and use an English sentiment analysis system. However, it is widely believed that aspects of sentiment may be lost in translation, especially in automatic translation. Though, the extent of this loss, in terms of drop in accuracy of automatic sentiment systems remains undetermined.

This chapter analyzes several methods available in annotating non-English texts for sentiment:

- Use a source-language sentiment analysis system.
- Run an English sentiment analysis system on manually created English translations of source language text.
- Run an English sentiment analysis system on automatically generated English translations of source language text.

In our experiments, we use Arabic social media posts as a specific instance of the source language text. We use state-of-the-art Arabic and English sentiment analysis systems as well as a state-of-the-art Arabic-to-English translation system. We outline the advantages and disadvantages of each of the methods listed above, and more importantly conduct experiments to determine accuracy of sentiment labels obtained using each of these methods. As benchmarks we use manually and automatically determined sentiment labels of the Arabic tweets.

These results will help users determine methods best suited for their particular needs. Along the way, we answer several research questions such as:

1. What sentiment prediction accuracy is expected when Arabic blog posts and tweets are translated into English (using the current state-of-art techniques), and then run through a state-of-the-art English sentiment analysis system?
2. How does this performance compare with that of a current state-of-the-art Arabic sentiment system?
3. What is the loss in sentiment predictability when translating Arabic text into English automatically vs. manually?
4. How difficult is it for humans to determine sentiment of automatically translated text?
5. When dealing with translated text, which is more accurate at determining the sentiment of Arabic text: (1) automatic sentiment analysis of the translated text, or (2) human annotation of the translated text for sentiment?

The inferences drawn from these experiments do not necessarily apply to language pairs other than Arabic–English. Languages can differ significantly in terms of characteristics that impact accuracy of an automatic sentiment analysis system. Our goal here specifically is to understand sentiment predictability of Arabic dialectal text on translation. However, a similar set of experiments can be used for other language pairs as well to determine the impact of translation on sentiment.

Through our experiments on two different datasets, we show that sentiment analysis of English translations of Arabic texts produces competitive results, w.r.t. Arabic sentiment analysis. We also show that translation (both manual and automatic) introduces marked changes in sentiment carried by the text; positive and negative texts can often be translated into texts that are neutral. We also find that certain attributes of automatically translated text that

mislead humans with regards to the true sentiment of the source text, do not seem to affect the automatic sentiment analysis system.

In the process of developing these experiments to study how translation alters sentiment, we created a state-of-the-art Arabic sentiment analysis system by porting NRC-Canada's competition winning system (Kiritchenko et al., 2014) to Arabic. We also created a substantial amount of sentiment labeled data pertaining to Arabic social media texts and their English translations which is made freely available.[1]

This is the first such resource where text in one language and its translations into another language (both manually and automatically produced) are each manually labeled for sentiment.

8.2 Related Work

8.2.1 Sentiment Analysis of English Social Media

Sentiment analysis systems have been applied to many different kinds of texts including customer reviews, newspaper headlines (Bellegarda, 2010), novels (Boucouvalas, 2002; Mohammad and Yang, 2011), emails (Liu et al., 2003; Mohammad and Yang, 2011), blogs (Neviarouskaya et al., 2011), and tweets (Mohammad, 2012). Often these systems have to cater to the specific needs of the text such as formality versus informality, length of utterances, etc. Sentiment analysis systems developed specifically for tweets include those by Go et al. (2009), Pak and Paroubek (2010), Agarwal et al. (2011), and Thelwall et al. (2011). A survey by Martínez-Cámara et al. (2012) provides an overview of the research on sentiment analysis of tweets. In the last two years, several shared tasks on sentiment analysis were organized by the Conference on Semantic Evaluation Exercises (SemEval), which allowed for comparison of different approaches on common datasets from different domains (Wilson et al., 2013; Rosenthal et al., 2014; Pontiki et al., 2014). The NRC-Canada system (Kiritchenko et al., 2014) ranked first in these competitions, and we use it in our experiments. Details of the system are described in Section 6.

[1] http://www.purl.com/net/ArabicSentiment

8.2.2 Sentiment Analysis of Arabic Social Media

Sentiment analysis of Arabic social media texts has several challenges. The text is often in a regional Arabic dialect rather than Modern Standard Arabic (MSA). Unlike MSA which is a standardized form of Arabic, dialectal Arabic is the spoken form of Arabic and lacks strict writing standards. The text often includes words from languages other than Arabic and multiple scripts may be used to express Arabic and foreign words. In addition, Arabic is a morphologically complex language, thus having a lexicon of word-sentiment associations that covers all different surface forms becomes a cumbersome task. Negation in MSA is expressed through negation particles, but in some dialects (Egyptian) it is expressed using suffixes at the end of the word. We refer the reader to Mourad and Darwish (2013) for more details on these issues.

There have been a few studies tackling sentiment analysis of Arabic texts (Ahmad et al., 2006; Badaro et al., 2014). The ones most closely related to our work are the studies of sentiment analysis of Arabic social media (Al-Kabi et al., 2013; El-Beltagy and Ali, 2013; Mourad and Darwish, 2013; Abdul-Mageed et al., 2014). Here we review existing Arabic sentiment analysis systems that were designed specifically for Arabic social media datasets. Abdul-Mageed et al. (2014) trained an SVM classifier on a manually labeled dataset and applied a two-stage classification that first separates subjective from objective sentences and then classifies the subjective into positive or negative instances. The authors have compiled several datasets from multiple social media resources that include chatroom messages, tweets, forum posts, and Wikipedia Talk pages. However, these resources have not been made publicly available yet.

Mourad and Darwish (2013) trained SVM and Naive Bayes classifiers on Arabic tweets annotated by two native Arabic speakers. We compare our system's performance to theirs in Section 7.

Refaee and Rieser (2014b) manually annotated tweets for sentiment by two native Arabic speakers. They used an SVM to classify tweets in a two-stage approach, polar vs neutral, then positive vs. negative. The authors shared

their data with us and we test our system on their dataset. However, the dataset they provided us is a larger superset than the one they had originally used (Refaee and Rieser, 2014a). Thus, the results of sentiment systems on the two sets are not directly comparable.

8.2.3 Multilingual Sentiment Analysis

Work on multilingual sentiment analysis has mainly addressed mapping sentiment resources from English into morphologically complex languages. Mihalcea et al. (2007) used English resources to automatically generate a Romanian subjectivity lexicon using an English–Romanian dictionary. The generated lexicon is then used to classify Romanian text. Wan (2008) translated Chinese customer reviews to English using a machine translation system. The translated reviews are then classified with a rule-based system that relies on English lexicons. A higher accuracy is achieved by using ensemble methods and combining knowledge from Chinese and English resources. Balahur and Turchi (2014) conducted a study to assess the performance of statistical sentiment analysis techniques on machine-translated texts. Opinion-bearing phrases from the *New York Times* text corpus (2002–2005) were automatically translated using publicly available machine-translation engines (Google, Bing, and Moses). Then, the accuracy of a sentiment analysis system trained on original English texts was compared to the accuracy of the system trained on automatic translations to German, Spanish, and French. The authors concluded that the quality of machine translation is sufficient for sentiment analysis to be performed on automatically translated texts without a substantial loss in accuracy. Contrary to that work, our study uses both manual and automatic translations as well as both manual and automatic sentiment assignments to systematically examine the effect of translation on sentiment. Additionally, we deal with noisy social media texts as opposed to more polished news media texts. There exists research on using sentiment analysis to improve machine translation (Chen and Zhu, 2014), but that is beyond the scope of this thesis.

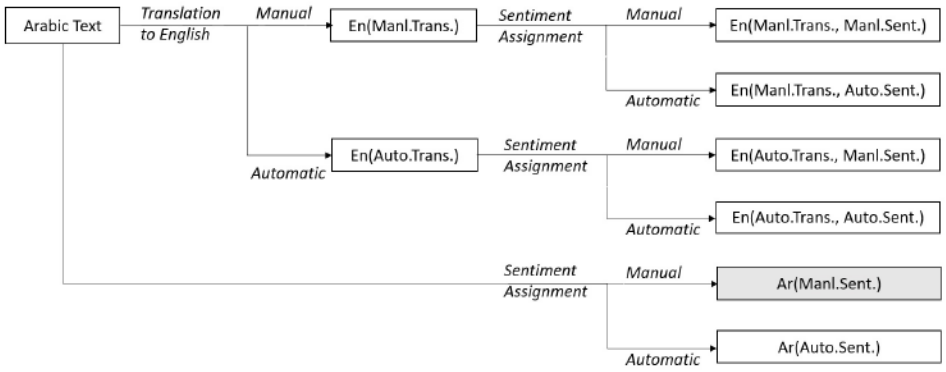


Figure 8.1: Experimental setup to determine the impact of translation on sentiment. We compare sentiment labels between Ar(Manl.Sent.) (shown in a shaded box) and other datasets shown on the right side of the figure. Ar(Manl.Sent.) is the original Arabic text manually annotated for sentiment.

8.3 Method for Determining Sentiment Predictability on Translation

In order to systematically study the impact of translation on sentiment analysis, we propose the following experimental setup:

- Identify or compile an Arabic social media dataset. We will refer to it as *Ar*. (*Ar* comes from the first two letter of Arabic.)
- Manually translate *Ar* into English. We will refer to these English translations as *En(Manl.Trans.)* [*Manl.* is for manual, and *Trans.* is for translations.]
- Automatically translate *Ar* into English. We will refer to these English translations as *En(Auto.Trans.)* [*Auto.* is for automatic.]
- Manually annotate Ar. for sentiment. We will refer to the sentiment-labeled dataset as *Ar(Manl.Sent.)*
- Manually annotate all English datasets [*En(Manl.Trans.)* and *En(Auto.Trans.)*] for sentiment, creating *En(Manl.Trans., Manl.Sent.)* and *En(Auto.Trans., Manl.Sent.)*, respectively.
- Run a state-of-the-art Arabic sentiment analysis system on *Ar*, creating *Ar(Auto.Sent.)*

- Run a state-of-the-art English sentiment analysis system on all the English datasets [*En(Manl.Trans.)* and *En(Auto.Trans.)*], creating *En(Manl. Trans., Auto.Sent.)* and *En(Auto.Trans., Auto.Sent.)*, respectively.

Figure 1 depicts this setup. Once the various sentiment-labeled datasets are created, we can compare pairs of datasets to draw inferences. For example, comparing the labels for *Ar(Manl.Sent.)* and *En(Manl.Trans., Manl.Sent.)* will show how different the sentiment labels tend to be when text is translated from Arabic to English. The comparison will also show, for example, whether positive tweets tend to often be translated into neutral tweets, and to what extent. The results will also show how feasible it is to first translate Arabic text into English and then use automatic sentiment analysis (*Ar(Manl.Sent.)* vs. *En(Auto.Trans., Auto.Sent.)*). In Section 8.8, we provide an analysis of several such comparisons for two different Arabic social media datasets.

DATA: Since manual translation of text from Arabic to English is a costly exercise, we chose, for our experiments, an existing Arabic social media dataset that has already been translated – the BBN Arabic-Dialect/English Parallel Text (Zbib et al., 2012).[2] It contains about 3.5 million tokens of Arabic dialect sentences and their English translations. We use a randomly chosen subset of 1200 Levantine dialectal sentences, which we will refer to as the *BBN posts* or *BBN dataset*, in our experiments. Additionally, we also conduct experiments on a dataset of 2000 tweets originating from Syria (a country where Levantine dialectal Arabic is commonly spoken). These tweets were collected in May 2014 by polling the Twitter API. We will refer to this dataset as the *Syrian tweets* or *Syrian dataset*. Note, however, that manual translations of the Syrian dataset are not available.

The experimental setup described above involves several component tasks: generating translations manually and automatically (Section 8.4), manually annotating Arabic and English texts for sentiment (Section 8.5), automatic sentiment analysis of English texts (Section 8.6), and automatic sentiment analysis of Arabic texts (Section 8.7).

[2]https://catalog.ldc.upenn.edu/LDC2012T09

8.4 Generating English Translations

The BBN dialectal Arabic dataset comes with manual translations into English. We generate automatic translations of the Arabic BBN posts and the Syrian tweets, by training a multi-stack phrase-based machine translation system to translate from Arabic to English. Our in-house system is quite similar to Cherry and Foster (2012). This statistical machine translation (SMT) system is trained on data from OpenMT 2012. We preprocess the training data by segmenting the Arabic source side of the training data with MADA 3.2 (Habash et al., 2009), using Penn Arabic Treebank (PATB) segmentation scheme as recommended by El Kholy and Habash (2012a). The Arabic script is further normalized by converting different forms of Alif إ أ آ ا and Ya ي ى to bare Alif ا and dotless Ya ى. The different forms are used interchangeably, and normalization decreases the sparcity of Arabic tokens and improves translation. The English side of the training data is lower-cased and tokenized by stripping punctuation marks. We set the decoder's stack size to 10000 and distortion limit to 7. We replace the *out-of-vocabulary* words in the translated text with *UNKNOWN* token (which is shown to the annotators). The decoder's log-linear model is tuned with MIRA (Chiang et al., 2008; Cherry and Foster, 2012). A KN-smoothed 5-gram language model is trained on the English Gigaword and the target side of the parallel data.

8.5 Creating sentiment labeled data in Arabic and English

Manual sentiment annotations were performed on the crowdsourcing platform CrowdFlower[3] for three BBN datasets and two Syrian datasets:

1. Original Arabic posts (BBN and Syria datasets), annotated by Arabic speakers.
2. Manual English translations of Arabic posts, annotated by English speakers (only for BBN dataset).

[3]http://www.crowdflower.com

	positive	negative	neutral	agreement
BBN data				
a. Ar(Manl.Sent)	41.50	47.92	10.58	73.80
b. En(Manl.Trans., Manl.Sent)	35.00	43.25	21.75	68.00
c. En(Auto.Trans., Manl.Sent)	36.17	36.50	27.34	65.70
Syria data				
d. Ar(Manl.Sent)	22.40	67.50	10.10	79.00
e. En(Auto.Trans., Manl.Sent)	14.25	66.15	19.60	76.10

Table 8.1: Class distribution (in percentage) of the sentiment annotated datasets.

3. Automatic English translations of Arabic posts (BBN and Syria datasets), annotated by English speakers.

Each post was annotated by at least ten annotators and the majority sentiment label was chosen. Table 8.1 shows the class distribution of sentiment labels in various datasets. Observe from rows *a* and *d* that neutral tweets constitute only about 10% of the data in both BBN and Syria datasets. The Syrian tweets have a much higher percentage of negative posts, whereas in the BBN data, the percentages of positive and negative posts are comparable. (Arabic tweets in general tend to be much more skewed to the negative class than Arabic blog post sentences.) Rows b, c, and e show that translated texts tend to lose some of the sentiment information and there is a relatively higher percentage of neutral instances in the translated text than in the original text.

For each post, we determine the count of the most frequent annotation divided by the total number of annotations. This score is averaged for all posts to determine the inter-annotator agreement shown in the last column of Table 8.1. We use this agreement score as benchmark to compare performance of automatic sentiment systems (described below).

8.6 English Sentiment Analysis

We use the English-language sentiment analysis system developed by NRC-Canada (Kiritchenko et al., 2014) in our experiments. This system obtained highest scores in two recent international competitions on sentiment analysis of tweets –SemEval-2013 Task 2 and SemEval-2014 Task 9 (Wilson et al., 2013; Rosenthal et al., 2014). We briefly describe the system below; for more details,

we refer the reader to Kiritchenko et al. (2014).

A linear-kernel Support Vector Machine (Chang and Lin, 2011) classifier is trained on the available training data. The classifier leverages a variety of surface-form, semantic, and sentiment lexicon features described below. The sentiment lexicon features are derived from existing, general-purpose, manual lexicons, namely NRC Emotion Lexicon (Mohammad and Turney, 2010, 2013), Bing Liu's Lexicon (Hu and Liu, 2004), and MPQA Subjectivity Lexicon (Wilson et al., 2005), as well as automatically generated, tweet-specific lexicons, Hashtag Sentiment Lexicon and Sentiment140 Lexicon (Kiritchenko et al., 2014).[4]

8.6.1 Generating English Sentiment Lexicon

Ablation experiments in Mohammad et al. (2013) showed that their sentiment system benefited most from the use of the Hashtag Sentiment Lexicon. The lexicon was created as follows. A list of 77 seed words, which are synonyms of *positive* and *negative*, was compiled from the Roget's Thesaurus. Then, the Twitter API was polled to collect tweets that had these words as hashtags. A tweet is considered positive if it has a positive hashtag and negative if it has a negative hashtag. For each term in the tweet set, a sentiment score is computed by measuring the PMI (pointwise mutual information) between the term and the positive and negative categories:

$$SenScore\,(w) = PMI(w, pos) - PMI(w, neg) \tag{8.1}$$

where w is a term in the lexicon. $PMI(w, pos)$ is the PMI score between w and the positive class, and $PMI(w, neg)$ is the PMI score between w and the negative class. A positive $SenScore\,(\mathrm{w})$ suggests that the word is associated with positive sentiment and a negative score suggests that the word is associated with negative sentiment. The magnitude indicates the strength of the association.

[4]http://www.purl.com/net/lexicons

Arabic Sentiment Dataset	**MD**	**RR**	**BBN**	**Syria**
sentiment classes	pos, neg	pos,neg	pos, neg, neu	pos, neg, neu
number of instances	1111	2681	1199	2000
Most frequent class baseline	66.06	68.92	47.95	67.50
Human agreement benchmark	-	-	73.82	79.05
MD Arabic SA system	72.50	-	-	-
Our Arabic SA system(surface)	70.65	84.0	62.0	**79.45**
Our Arabic SA system(lemma)	**74.62**	**85.23**	**63.89**	78.96

Table 8.2: Accuracy (in percentage) of sentiment analysis (SA) systems on various Arabic social media datasets. Accuracy is calculated based of sum of true positives of all folds.

8.6.2 Pre-processing and Feature Generation

The following pre-processing steps are performed. URLs and user mentions are normalized to http://someurl and @someuser, respectively. Tweets are tokenized and part-of-speech tagged with the CMU Twitter NLP tool (Gimpel et al., 2011). Then, each tweet is represented as a feature vector.

The features:

- Word and character ngrams;

- POS: # occurrences of each part-of-speech tag;

- Negation: # negated contexts. Negation also affects the ngram features: a word w becomes w_NEG in a negated context;

- Automatic sentiment lexicons: For each token w occurring in a tweet, its sentiment score $score(w)$ is used to compute: # tokens with $score(w) \neq 0$; the total score $= \sum_{w \in tweet} score(w)$; the maximal score $= max_{w \in tweet} score(w)$; the score of the last token in the tweet.

- Manually created sentiment lexicons: For each of the three manual sentiment lexicons, the following features are computed: the sum of positive and the sum of negative scores for tweet tokens in affirmative contexts and in negated contexts, separately.

	surface		lemma	
	count	**coverage**	**count**	**coverage**
BBN	5897	56%	3885	67%
Syria	9856	63%	5644	73%
RR	16005	58%	8416	66%
MD	7620	65%	4706	73%

Table 8.3: Type counts for surface and lemmatized forms of the training sets. The coverage is the percentage of types in the training data that is covered by the hashtag lexicon.

8.7 Arabic Sentiment Analysis

8.7.1 Building an Arabic Sentiment System

We built an Arabic sentiment analysis system by reconstructing the NRC-Canada English system to deal with Arabic text. It extracts all of the feature described in Section 8.6.2 except POS and negation features. We also generated a word-sentiment association lexicon as described in Section 8.6.1, but for Arabic words from Arabic tweets (more details in sub-section below). We preprocess Arabic text by tokenizing with CMU Twitter NLP tool to deal with specific tokens such as URLs, usernames, and emoticons. Then we use MADA to generate lemmas. Finally, we normalize different forms of Alif and Ya to bare Alif and dotless Ya to decrease token sparcity in Arabic datasets.

Generating Arabic Sentiment Lexicon

We translated 77 positive and negative seed words used to generate the English NRC Hashtag Sentiment Lexicon into Arabic using Google Translate. Among the several translations provided by it, we chose words that were less ambiguous and tended to have strong sentiment in Arabic texts. To increase the coverage of our seed list, we manually added different inflections for these translations.

We polled the Twitter API for the period of June to August 2014 and collected tweets with *#(keyword)*. After filtering out duplicate tweets and retweets, we ended up with 163,944 positive unique tweets and 37,848 negative unique tweets. All tweets are preprocessed by normalizing the Alif and Ya

	pos	neg	neu
BBN data			
a. Ar(Auto.Sent)	39.78	60.05	0.17
b. En(Manl.Trans., Auto.Sent)	43.12	55.63	1.25
c. En(Auto.Trans., Auto.Sent)	42.87	56.05	1.08
Syria data			
d. Ar(Auto.Sent)	20.60	75.30	4.10
e. En(Auto.Trans., Auto.Sent)	24.75	69.75	5.50

Table 8.4: Class distribution (in percentage) resulting from automatic sentiment analysis.

characters and converting words to their lemmatized form. Then for each word w, $SenScore\,(\mathrm{w})$ was calculated just as described in Section 8.6.1.

8.7.2 Evaluation

We tested the Arabic sentiment system on two existing Arabic datasets (Mourad and Darwish (2013) (MD) and Refaee and Rieser (2014a) (RR)) and two newly sentiment-annotated Arabic datasets (BBN and Syria). Table 8.2 shows results of ten-fold cross-validation experiments on each of the datasets. For MD and RR, the presented results are for the two-class problem (positive vs. negative) to allow for comparison with prior published results. For BBN and Syria, the results are shown for the case where the system has to identify one of three classes: positive, negative, or neutral. Human agreement scores are shown where available.

We experimented using 2 versions of our system, one trained on the surface forms of the training data and another one trained on their lemmas. Table 8.2 shows that results from training on lemma forms surpasses the ones on the original surface forms. Lemmatization decreases the lexical sparsity mainly through dropping out the affixes from the words and converting it to a form that maintains its core meaning. Table 8.3 shows a decrease in token types by around 40% for these datasets. We also calculated the percentage of types for each dataset that is covered by the hashtag lexicon. A lemmatized lexicon results with higher coverage to terms in the datasets compared to one generated from surface forms.

Note that the accuracy of our system (trained on lemmas) is higher than

previously published results on the MD dataset. The only previously published results on the RR dataset are on a small subset (about 1000 instances) for which Refaee and Rieser (2014a) obtained an accuracy of 87%. The results in Table 8.2 are for a larger dataset and so not directly comparable.

8.8 Sentiment After Translation

Using the methods and systems described in Sections 8.4, 8.5, 8.6, and 8.7, we generated all the manually and automatically labeled datasets mentioned in Section 8.3's Experimental Setup. Table 8.4 shows the distribution of positive, negative, and neutral classes in datasets that have been automatically labeled with sentiment. These percentages can be compared with those in Table 8.1 (rows a and d) which show the true sentiment distribution in the BBN and Syria datasets. Observe that the automatic system has difficulty in assigning neutral class to posts. This is probably because of the small percentage (about 10%) of neutral tweets in the training data. Also notice that the system predominantly guesses negative, which is also a reflection of the distribution in the training data. The strong bias to negatives is lessened in the English translations.

Main Result: Tables 8.5 and 8.6 show how similar the sentiment labels are across various pairs of datasets for the BBN posts and the Syrian posts, respectively. For example, row a. in Table 8.5 shows the comparison between Arabic tweets that were manually annotated for sentiment and those that were automatically labeled for sentiment by our Arabic sentiment analysis system. Column 2 shows the percentage of instances where the sentiment labels match across the two datasets being compared. For row a. the match percentage of 63.89% represents the accuracy of the automatic sentiment analysis system on the Arabic BBN posts.

Row b. shows the difference in labels when text is manually translated from Arabic to English, even though sentiment labeling in both Arabic and English is done manually. Observe that the two labels match only 71.31% of the time. However, the agreement among human sentiment annotators on

Data Pair	Match %
a. Ar(Manl.Sent) - Ar(Auto.Sent)	63.89
b. Ar(Manl.Sent) - En(Manl.Trans., Manl.Sent)	71.31
c. Ar(Manl.Sent) - En(Manl.Trans., Auto.Sent)	68.65
d. Ar(Manl.Sent) - En(Auto.Trans., Manl.Sent)	57.21
e. Ar(Manl.Sent) - En(Auto.Trans., Auto.Sent)	62.49
f. En(Manl.Trans., Manl.Sent) - En(Auto.Trans., Manl.Sent)	60.08
g. En(Manl.Trans., Manl.Sent) - En(Manl.Trans., Auto.Sent)	66.51
h. En(Auto.Trans., Manl.Sent) - En(Auto.Trans., Auto.Sent)	69.58

Table 8.5: Match percentage between pairs of sentiment labelled BBN datasets.

Data Pair	Match %
a. Ar(Manl.Sent) - Ar(Auto.Sent)	78.96
b. Ar(Manl.Sent) - En(Auto.Trans., Manl.Sent)	71.05
c. Ar(Manl.Sent) - En(Auto.Trans.-Auto.Sent)	78.11
d. En(Auto.Trans, Manl.Sent) - En(Auto.Trans., Auto.Sent)	78.80

Table 8.6: Match percentage between pairs of sentiment labelled Syria datasets.

original Arabic texts was only 73.8%. So, the English translation does affect sentiment, but not dramatically.

Row c. shows results for when the manually translated text is run through an English sentiment analysis system and the labels are compared against *Ar(Manl.Sent.)* Observe that the match for this pair is 68.65%, which is not too much lower than 71.31% obtained by manual sentiment labeling. This shows that the English sentiment system is performing rather well. (One would not expect it to get a match greater than 71.31%.) More importantly, the English sentiment system shows a competitive result of 62.49% when run on the automatically translated text (row e.), which makes this choice a viable option for sentiment analysis of non-English texts. This result is inline with previous findings in Information Retrieval (Nie et al., 1999) and Text Classification (Amini and Goutte, 2010).

Rows d. and e. compare *Ar(Manl.Sent.)* with manual and automatic sentiment labeling of automatic translations. Since automatic translation from Arabic to English is fairly difficult, we expect these match percentages to be lower than those in rows b. and c., and that is exactly what we observe. However, it is unexpected to find the number for row e. to be higher than that of

row d. We find the same pattern for corresponding data pairs in the Syrian tweets as well (rows b. and c. in Table 6). This suggests that certain attributes of automatically translated text mislead humans with regards to the true sentiment of the source text. However, these same attributes do not seem to affect the automatic sentiment analysis system as much. Since the NRC sentiment analysis system is largely reliant on word-sentiment associations and does not use syntax-based features, it is possible that syntactic abnormalities introduced by automatic translation impact human perception of sentiment.

Row f. shows that manual and automatic translation lead to only about 60% match in manually annotated sentiment labels with each other. Row g. shows accuracy of the English automatic sentiment analysis system on the manually translated text (assuming the English sentiment labels as gold). The result of 66.51% is very close to human agreement on manually translated data (68%), which demonstrates the high quality of the English sentiment analysis system. Row h. shows accuracy of the English automatic sentiment analysis system on the automatically translated text (assuming the English sentiment labels as gold). In this case, the system's accuracy of 69.58% is higher than the human agreement on automatically translated text (65.7%), which again shows that automatic translation greatly impacts sentiment perceived by humans.

We manually examined several tweets from the BBN dataset to understand why humans incorrectly annotate a tweet's automatic translation. Most of the cases were due to bad translation where sentiment words either disappeared or were replaced with words of opposite sentiment. In some cases, the translation was affected by typos on the Arabic side. Table 8.7 shows some examples. Often the mistranslations occurred due to word sense ambiguity. For example, عقارب has two meanings: *scorpions* and *clock arms*. In example 1 (metaphorically stating that relatives can hurt like scorpion bites), the word is mistranslated, leading to neutral (instead of negative) sentiment.

One reason why the automatic sentiment analysis system correctly annotates several automatically translated instances (where manual annotations of the translation may fail), is that the system can learn an appropriate model even from mistranslated text — especially when automatic translation makes

1. Bad auto. translation: mistranslation of ambiguous words		
Post	الدنيا علمتني ان اكثر الاقارب عقارب	negative
Auto.Trans.	the minimum taught me that more relatives clock	neutral
Manl.Trans.	Life has taught me that most of the relatives are scorpions	negative
2. Bad auto. translation: mistranslation of ambiguous words		
Post	ليتنى اعيش في مكان لا تنقطع عنه الثلوج	positive
Auto.Trans.	i wish i live in a place not cut off by snow	negative
Manl.Trans.	I wish I live in a place where snow never stops falling	positive
3. Bad auto. translation: sarcasm is hard to translate		
Post	لسه الخير لقدام تسرب المي موجودة من زمان	negative
Auto.Trans.	you're still good in front of the leakage of water existed from time	positive
Manl.Trans.	Expect more good to come, water has been leaking since a long time	negative

Table 8.7: Examples where the automatic translation was annotated a sentiment different from the sentiment of the original Arabic tweet, but whose original sentiment was correctly predicted by the English sentiment system. The manual translations are also listed for reference.

consistent errors. For example, اللهم انصر (*Oh God grant victory to*) has been consistently translated to *God forsake*. All tweets having this phrase are correctly annotated as positive by our system, but were marked negative by the human annotators.

Caveats: The automatic systems employed in these experiments, i.e., Arabic sentiment analysis, English sentiment analysis, and SMT systems, exhibit state-of-the-art performance; nevertheless, further improvements are possible. The Arabic sentiment system will benefit from extended sentiment lexicons and features derived specifically for the Arabic language. The English sentiment analysis system can be further adapted to the peculiarities of machine-translated texts, which are notably different from regular English. The current translation system has been trained on non-tweet data that results in a high percentage of out-of-vocabulary words on our datasets. In our experiments, we assumed that all texts are written in Levantine dialect of the Arabic language. However, tweets can have a mixture of dialects or even a mixture of languages (e.g., Arabic and English). Addressing these factors will give even more insight on how sentiment is altered on translation, in specific contexts.

8.9 Summary

We presented a set of experiments to systematically study the impact of English translation (manual and automatic) on sentiment analysis of Arabic social media posts. Our experiments show that automatic sentiment analysis of English translations (even of automatic translations) can lead to competitive results—results that are similar to that obtained by current state-of-the-art Arabic sentiment analysis systems. Our results also show that automatic sentiment analysis of automatic translations outperforms the manual sentiment annotations of the automatically translated text. This suggests that SMT errors impact human perception of sentiment markedly more than automatic sentiment systems. We also show that translated texts tend to lose some of the sentiment information and there is a relatively higher percentage of neutral instances in the translated text than in the original dataset. The resources created as part of this project (Arabic sentiment lexicons, Arabic sentiment annotations of social media posts, and English sentiment annotations of their translations) are made freely available.[5]

[5]http://www.purl.com/net/ArabicSentiment

Chapter 9

Conclusion

Morphological segmentation and morphological analysis are effective approaches for sparsity reduction in morphologically complex languages, specifically Arabic. Segmentation manages to improve the SMT system by improving correspondence between source and target language and decreasing token sparsity. Similarly, lemmatization improves sentiment analysis through maintaining core meaning of the word while reducing it to a simpler form. In this dissertation, we aimed to improve translation from English to Arabic in a phrase-based SMT framework. We addressed several issues concerning desegmentation and moved it from post-processing to a decoder-integrated process. This opens an opportunity to benefit from both morpheme-based and word-based properties of the Arabic language. We also measured sentiment predictability when Arabic social media text is translated to English manually and automatically, and compared the results to ones obtained using Arabic sentiment analysis system. In this final chapter, we summarize our work in this dissertation, highlight the main contributions and discuss future work.

9.1 Summary

We provided a desegmentation technique that is language-independent and overcomes the limitations of available techniques. We approach desegmentation as a string transduction problem. Our DIRECTL+ tool is trained on aligned segmented and unsegmented forms where character aligned pairs become the operation of transduction. When applied to a naturally occurring

Arabic text, our approach performs the best with a word error rate of 0.087. However, when tested on a segmented Arabic SMT output, we get similar results to a Table+Rule-based technique. Our results suggested that merely improving the desegmentation technique might not contribute much to the improvement in translation quality. Since desegmentation is dealt with as a post-processing step, propagated decoder errors are difficult to overcome. Our analysis implies that the benefits of desegmentation could be well-utilized at earlier points of the SMT pipeline.

In Chapter 5, we provide an algorithm that desegments the search space built from segmented Arabic tokens. This supports the SMT system with two views of the search space. Initially, the system has a morpheme-based view, where the model's features are based on segmented Arabic. The second view is word-based view, where the lattice is desegmented. The desegmented lattice enables extraction of word-level features, such as scoring on an unsegmented language model or using discontiguity features. Our results shows an improvement of 0.7 BLEU points compared to the 1-best-desegmentation baseline. The system seems to have two main benefits from desegmentation and adding word-level features. First, the desegmented form's correctness is always evaluated with the unsegmented language model, given its desegmented context. The unsegmented language model can span a larger morphemic context compared to segmented language model. Second, our analysis reveals that discontiguity features have a positive effect on the choice of clitics. Desegmented words aligned to consecutive source tokens are favored by the system over ones aligned to discontiguous source tokens. An incorrect clitic in a desegmented form can not contribute to the BLEU score. Among several correct translations of English prepositions, our system is able to choose the correct ones that contributes to the BLEU score.

Our lattice desegmentation algorithm showed significant improvement in the quality of the translation. Yet, desegmenting the lattice is not the only option. In Chapter 6, we provide a systematic study that explores all options of integrating desegmentation in the SMT pipeline by changing the point where we apply desegmentation. In addition to the lattice, we apply desegmentation

to the alignment and the phrase table. One thing these approaches have in common is that they all output unsegmented Arabic, but exploit segmentation properties. Our results confirmed that lattice desegmentation was the best integration option. By desegmenting the alignment and phrase table, we are giving up the most significant property of desegmentation: phrases with flexible boundaries. Although these phrases accounted for 12% of phrases in the output, they appear to be responsible for the difference in BLEU score between the one-best-desegmentation and the unsegmented baseline. We also examined the impact of an unsegmented language model in our experiments. The unsegmented language models always contributed to the BLEU score. But at the same time, it caused a drop in the use of morphologically complex words. Our analysis suggests that systems using unsegmented language models could also be improved by using features that encourage the morphological productivity of the system, as this can have a positive impact on fluency.

A limitation of our lattice desegmentation approach is that it applies desegmentation as an offline process to a pre-generated search space from the decoder. To overcome this limitation, we developed a model that integrates desegmentation directly with the decoder (Chapter 7). Desegmentation and word-level feature extraction are handled while hypotheses are expanded in the decoder. Our in-decoder approach produces results with similar scores to the lattice desegmentation. At the same time, the method adds no complications to the SMT pipeline, which remains as simple as train, tune and decode (while the desegmentation is running through the decoding).

On a different work, we managed to evaluate the loss of sentiment predictability when Arabic tweets are translated manually and automatically into English (Chapter 8). Competitive results were shown compared to running our state-of-the-art Arabic sentiment analysis system developed by us on Arabic tweets. Also, sentiment analysis on automatically translated text surpasses the human annotation of automatically translated text. Our analysis reveals that the SMT system translates Arabic source tokens more consistently. Since the automatic sentiment analysis system is trained on these consistently translated text with the original sentiment labels of the source text, it is still able

to determine the true sentiment. However, since human sentiment annotators see many instances where the sentiment terms are mistranslated into neutral terms, they are unable to determine the true sentiment.

9.2 Future Work

The desegmentation technique we adopt in Chapters 5, 6 and 7 is table-based. As desegmentation can lead to different unsegmented word forms, we limited our choice to the most frequent option from the desegmentation table. We plan to expand this work and handle multiple desegmentation options. Also, we plan to experiment with translating from English into Arabic lemmas. The purpose of the table in this case is to map sequences of Arabic lemmas with their clitics to an inflected form attached to its clitics. This increases the options in our search space. Moreover, we can leave choosing the better option to the unsegmented language model, or to integrating additional word-based or part-of-speech features.

The unsegmented language model in our in-decoder desegmentation approach in Chapter 7 can not handle out of context scoring of phrases with flexible boundaries. We plan to improve the unsegmented language modeling such that it can give a better estimation for such cases. This allows good hypotheses to be still considered in the search space, and not get discarded early.

In this dissertation, we handled the problems of translation from English into Arabic from a morphological perspective. An interesting future plan is to expand this work to also handle agreements that affect the Arabic words' surface forms. A language model usually handles agreement implicitly. However, when translating into Arabic with its rich morphology, we need to integrate more information about the context. In particular, syntactic features can be incorporated into the system, which can be passed to future translation options.

A large number of out-of-vocabulary words are proper nouns that are usually left untranslated or dropped from the final output. The SMT system can

be improved by transliterating these words. Our research on transliterating Arabic names to English is a state-of-the-art (Kondrak et al., 2012; Nicolai et al., 2015). We plan to train transliteration from English to Arabic and integrate the transliteration system with the SMT system to better handle such transliterations.

Our current Arabic sentiment analysis system can not handle negations terms in Arabic. Negation in modern standard Arabic (MSA) appears as a free morpheme, while it appears as a circumfix in some dialects. One interesting direction is to allow the system to handle negations by taking into consideration the scope that it negates.

Bibliography

Abdul-Mageed, M., Diab, M., and Kübler, S. (2014). SAMAR: Subjectivity and sentiment analysis for Arabic social media. *Computer Speech & Language*, 28(1):20 – 37.

Agarwal, A., Xie, B., Vovsha, I., Rambow, O., and Passonneau, R. (2011). Sentiment analysis of Twitter data. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 30–38, Portland, Oregon.

Ahmad, K., Cheng, D., and Almas, Y. (2006). Multi-lingual sentiment analysis of financial news streams. In *Proceedings of the 1st International Conference on Grid in Finance*.

Al-Kabi, M., Gigieh, A., Alsmadi, I., Wahsheh, H., and Haidar, M. (2013). An opinion analysis tool for colloquial and standard Arabic. In *Proceedings of the 4th International Conference on Information and Communication Systems*, ICICS '13.

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., and Mohri, M. (2007). OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer. `http://www.openfst.org`.

Amini, M.-R. and Goutte, C. (2010). A co-classification approach to learning from multilingual corpora. *Machine learning*, 79(1-2):105–121.

Badaro, G., Baly, R., Hajj, H., Habash, N., and El-Hajj, W. (2014). A large scale Arabic sentiment lexicon for Arabic opinion mining. In *Proceedings of the EMNLP Workshop on Arabic Natural Language Processing (ANLP)*, pages 165–173, Doha, Qatar.

Badr, I., Zbib, R., and Glass, J. (2008). Segmentation for English-to-Arabic statistical machine translation. In *Proceedings of ACL*, pages 153–156.

Balahur, A. and Turchi, M. (2014). Comparative experiments using supervised learning and machine translation for multilingual sentiment analysis. *Computer Speech & Language*, 28(1):56–75.

Bellegarda, J. (2010). Emotion analysis using latent affective folding and embedding. In *Proceedings of the NAACL-HLT Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 1–9, Los Angeles, California.

Bojar, O. (2007). English-to-Czech factored machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 232–239, Prague, Czech Republic.

Boucouvalas, A. C. (2002). Real time text-to-emotion engine for expressive Internet communication. *Emerging Communication: Studies on New Technologies and Practices in Communication*, 5:305–318.

Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79–85.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Buckwalter, T. (2004). Buckwalter arabic morphological analyzer (bama) version 2.0. Technical report, Linguistic Data Consortium LDC2004L02.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27.

Chen, B. and Zhu, X. (2014). Bilingual sentiment consistency for statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 607–615, Gothenburg, Sweden. Association for Computational Linguistics.

Cherry, C. and Foster, G. (2012). Batch tuning strategies for statistical machine translation. In *Proceedings of HLT-NAACL*, Montreal, Canada.

Chiang, D., Marton, Y., and Resnik, P. (2008). Online large-margin training of syntactic and structural translation features. In *Proceedings of EMNLP*, pages 224–233.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of ACL*, pages 176–181.

Clifton, A. and Sarkar, A. (2011). Combining morpheme-based machine translation with post-processing morpheme prediction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 32–42, Portland, Oregon, USA.

Creutz, M. and Lagus, K. (2005). Inducing the morphological lexicon of a natural language from unannotated text. In *In Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRRâĂŹ05*, pages 106–113.

Dyer, C., Muresan, S., and Resnik, P. (2008). Generalizing word lattice translation. In *Proceedings of ACL-08: HLT*, pages 1012–1020, Columbus, Ohio.

El-Beltagy, S. R. and Ali, A. (2013). Open issues in the sentiment analysis of Arabic social media: A case study. In *Proceedings of the 9th International Conference on Innovations in Information Technology*, pages 215–220. IEEE.

El Kholy, A. and Habash, N. (2012a). Orthographic and morphological processing for English—Arabic statistical machine translation. *Machine Translation*, 26(1-2):25–45.

El Kholy, A. and Habash, N. (2012b). Translate, predict or generate: Modeling rich morphology in statistical machine translation. *Proceeding of the Meeting of the European Association for Machine Translation.*

Fraser, A., Weller, M., Cahill, A., and Cap, F. (2012). Modeling inflection and word-formation in SMT. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 664–674, Avignon, France. Association for Computational Linguistics.

Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., and Smith, N. A. (2011). Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, ACL '11, pages 42–47.

Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. Technical report, Stanford University.

Green, S. and DeNero, J. (2012). A class-based agreement model for generating accurately inflected translations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–155, Jeju Island, Korea.

Habash, N. (2010). *Introduction to Arabic Natural Language Processing.* Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Habash, N., Rambow, O., and Roth, R. (2009). Mada+ tokan: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, pos tagging, stemming and lemmatization. In *Proceedings of the 2nd international conference on Arabic language resources and tools (MEDAR), Cairo, Egypt*, pages 102–109.

Habash, N., Roth, R., Rambow, O., Eskander, R., and Tomeh, N. (2013). Morphological analysis and disambiguation for dialectal arabic. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 426–432, Atlanta, Georgia.

Habash, N., Soudi, A., and Buckwalter, T. (2007). On arabic transliteration. In Soudi, A., Bosch, A. d., and Neumann, G., editors, *Arabic Computational Morphology*, volume 38 of *Text, Speech and Language Technology*, pages 15–22. Springer Netherlands.

Hoang, H., Birch, A., Callison-burch, C., Zens, R., Aachen, R., Constantin, A., Federico, M., Bertoldi, N., Dyer, C., Cowan, B., Shen, W., Moran, C., and Bojar, O. (2007). Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL), demonstration session*, pages 177–180.

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Huang, L. (2008). Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio.

Huang, L. and Chiang, D. (2007). Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic.

Hunt, J. W. and McIlroy, M. D. (1976). An algorithm for differential file comparison. Technical report, Bell Laboratories.

Jeong, M., Toutanova, K., Suzuki, H., and Quirk, C. (2010). A discriminative lexicon model for complex morphology. In *The Ninth Conference of the Association for Machine Translation in the Americas*.

Jiampojamarn, S., Cherry, C., and Kondrak, G. (2008). Joint processing and discriminative training for letter-to-phoneme conversion. In *Proceedings of ACL-08: HLT*, pages 905–913, Columbus, Ohio. Association for Computational Linguistics.

Jiampojamarn, S., Kondrak, G., and Sherif, T. (2007). Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York. Association for Computational Linguistics.

Kiritchenko, S., Zhu, X., and Mohammad, S. M. (2014). Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

Koehn, P. and Hoang, H. (2007). Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic. Association for Computational Linguistics.

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

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 127–133.

Kondrak, G., Li, X., and Salameh, M. (2012). Transliteration experiments on chinese and arabic. In *Proceedings of the 4th Named Entity Workshop*, pages 71–75. Association for Computational Linguistics.

Lee, Y. K., Haghighi, A., and Barzilay, R. (2011). Modeling syntactic context improves morphological segmentation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 1–9, Portland, Oregon, USA.

Liu, H., Lieberman, H., and Selker, T. (2003). A model of textual affect sensing using real-world knowledge. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, pages 125–132, New York, NY. ACM.

Luong, M.-T., Nakov, P., and Kan, M.-Y. (2010). A hybrid morpheme-word representation for machine translation of morphologically rich languages. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 148–157, Cambridge, MA.

Martínez-Cámara, E., Martín-Valdivia, M. T., Ureñalópez, L. A., and Montejoráez, A. R. (2012). Sentiment analysis in Twitter. *Natural Language Engineering*, pages 1–28.

Michael Denkowski, C. D. and Lavie, A. (2014). Learning from post-editing: Online model adaptation for statistical machine translation. In *Proceedings of EACL*.

Mihalcea, R., Banea, C., and Wiebe, J. (2007). Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, page 976.

Minkov, E., Toutanova, K., and Suzuki, H. (2007). Generating complex morphology for machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 128–135, Prague, Czech Republic.

Mohammad, S. M. (2012). #Emotional tweets. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, *SEM '12, pages 246–255, Montréal, Canada.

Mohammad, S. M., Kiritchenko, S., and Zhu, X. (2013). NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the 7th International Workshop on Semantic Evaluation Exercises*, SemEval '13, Atlanta, Georgia, USA.

Mohammad, S. M., Salameh, M., and Kiritchenko, S. (2016). How translation alters sentiment. *Journal of Artificial Intelligence Research*, 55:95–130.

Mohammad, S. M. and Turney, P. D. (2010). Emotions evoked by common words and phrases: Using Mechanical Turk to create an emotion lexicon. In *Proceedings of the NAACL-HLT Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34, LA, California.

Mohammad, S. M. and Turney, P. D. (2013). Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Mohammad, S. M. and Yang, T. W. (2011). Tracking sentiment in mail: How genders differ on emotional axes. In *Proceedings of the ACL Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA '11, pages 70–79, Portland, OR, USA.

Monroe, W., Green, S., and Manning, C. D. (2014). Word segmentation of informal arabic with domain adaptation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 206–211, Baltimore, Maryland.

Mourad, A. and Darwish, K. (2013). Subjectivity and sentiment analysis of modern standard Arabic and Arabic microblogs. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, WASSA '13, pages 55–64.

Neviarouskaya, A., Prendinger, H., and Ishizuka, M. (2011). Affect analysis model: novel rule-based approach to affect sensing from text. *Natural Language Engineering*, 17:95–135.

Nicolai, G., Hauer, B., Salameh, M., St Arnaud, A., Xu, Y., Yao, L., and Kondrak, G. (2015). Multiple system combination for transliteration. *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 72.

Nicolai, G., Hauer, B., Salameh, M., Yao, L., and Kondrak, G. (2013). Cognate and misspelling features for natural language identification. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 140–145.

Nie, J.-Y., Simard, M., Isabelle, P., and Durand, R. (1999). Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the Web. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–81. ACM.

Och, F. J. (2003). Minimum error rate training for statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Och, F. J., Ney, H., Josef, F., and Ney, O. H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29.

Oflazer, K. and Durgar El-Kahlout, I. (2007). Exploring different representational units in English-to-Turkish statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 25–32, Prague, Czech Republic.

Pak, A. and Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the 7th Conference on International Language Resources and Evaluation*, LREC '10, pages 1320–1326, Valletta, Malta.

Papineni, K., Roukos, S., Ward, T., and jing Zhu, W. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., and Manandhar, S. (2014). SemEval-2014 Task 4: Aspect based sentiment analysis. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '14, pages 27–35, Dublin, Ireland.

Refaee, E. and Rieser, V. (2014a). An Arabic Twitter corpus for subjectivity and sentiment analysis. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, LREC '14, Reykjavik, Iceland. European Language Resources Association.

Refaee, E. and Rieser, V. (2014b). Subjectivity and sentiment analysis of Arabic Twitter feeds with limited resources. In *Proceedings of the Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools*, page 16.

Roark, B., Sproat, R., and Shafran, I. (2011). Lexicographic semirings for exact automata encoding of sequence models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1–5, Portland, Oregon, USA.

Rosenthal, S., Ritter, A., Nakov, P., and Stoyanov, V. (2014). SemEval-2014 Task 9: Sentiment analysis in Twitter. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, SemEval '14, pages 73–80, Dublin, Ireland.

Sadat, F. and Habash, N. (2006). Combination of arabic preprocessing schemes for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Salameh, M., Cherry, C., and Kondrak, G. (2013). Reversing morphological tokenization in English-to-Arabic SMT. In *Proceedings of the 2013 NAACL HLT Student Research Workshop*, pages 47–53, Atlanta, Georgia.

Salameh, M., Cherry, C., and Kondrak, G. (2014). Lattice desegmentation for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 100–110.

Salameh, M., Cherry, C., and Kondrak, G. (2015a). What matters most in morphologically segmented smt models? *Syntax, Semantics and Structure in Statistical Translation*, page 65.

Salameh, M., Mohammad, S., and Kiritchenko, S. (2015b). Sentiment after translation: A case-study on Arabic social media posts. pages 767–777.

Sirts, K. and Goldwater, S. (2013). Minimally-supervised morphological segmentation using adaptor grammars. *TACL*, 1:255–266.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*.

Somers, H. (1992). An introduction to machine translation.

Stallard, D., Devlin, J., Kayser, M., Lee, Y. K., and Barzilay, R. (2012). Unsupervised morphology rivals supervised morphology for arabic mt. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 322–327, Stroudsburg, PA, USA.

Stolcke, A. (2002). Srilm - an extensible language modeling toolkit. In *Intl. Conf. Spoken Language Processing*, pages 901–904.

Subotin, M. (2011). An exponential translation model for target language morphology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238, Portland, Oregon, USA.

Thelwall, M., Buckley, K., and Paltoglou, G. (2011). Sentiment in Twitter events. *Journal of the American Society for Information Science and Technology*, 62(2):406–418.

Toutanova, K., Suzuki, H., and Ruopp, A. (2008). Applying morphology generation models to machine translation. In *Proceedings of ACL-08: HLT*, pages 514–522, Columbus, Ohio.

Ueffing, N., Och, F. J., and Ney, H. (2002). Generation of word graphs in statistical machine translation. In *Proceedings of EMNLP*, pages 156–163, Philadelphia, PA.

Wan, X. (2008). Using bilingual knowledge and ensemble techniques for unsupervised Chinese sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 553–561, Stroudsburg, PA, USA. Association for Computational Linguistics.

Weaver, W. (1949/1955). Translation. In Locke, W. N. and Boothe, A. D., editors, *Machine Translation of Languages*, pages 15–23. MIT Press, Cambridge, MA. Reprinted from a memorandum written by Weaver in 1949.

Wilson, T., Kozareva, Z., Nakov, P., Rosenthal, S., Stoyanov, V., and Ritter, A. (2013). SemEval-2013 Task 2: Sentiment analysis in Twitter. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '13, Atlanta, Georgia, USA.

Wilson, T., Wiebe, J., and Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA.

Zbib, R., Malchiodi, E., Devlin, J., Stallard, D., Matsoukas, S., Schwartz, R., Makhoul, J., Zaidan, O. F., and Callison-Burch, C. (2012). Machine translation of Arabic dialects. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 49–59. Association for Computational Linguistics.