Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Arabic Recognition and Translation System

Proyecto Final de Carrera

Ingeniería Informática

Autor: Ihab Alkhoury

Directores: Prof. Alfons Juan and Prof. Antonio Molina

September 23, 2013

Abstract

To our knowledge, there are only few systems that are able to automatically translate handwritten text images into another language, in particular, Arabic. Typically, the available systems are based on a concatenation of two systems: a Handwritten Text Recognition (HTR) system and a Machine Translation (MT) system. Roughly speaking, in the case of recognition of Arabic text images, our work has focused on the use of the embedded Bernoulli (mixture) HMMs (BHMMs), that is, embedded HMMs in which the emission probabilities are modeled with Bernoulli mixtures. In the case of Arabic text translation, our work has focused on one of the state-of-the-art phrase-based log-linear translation models. In this work we evaluate our system on the LDC corpus introduced in the NIST OpenHaRT 2010 and 2013 evaluations. Very competitive and promising results are shown. Additionally, we present the idea of a simple mobile application system for image translation that recognizes the Arabic text in an image and translates the recognized text into English.

Keywords: Arabic Recognition and Translation, NIST OpenHaRT, Bernoulli HMMs, Phrase-based models

Contents

Chapter 1

Introduction

To our knowledge, there are only few systems that automatically translate text images into another language, in particular, Arabic. They are usually based on statistical models in which Handwritten Text Recognition (HTR) is coupled with Machine Translation (MT) in order to translate text images. The whole process can be seen as a black box that takes as input a text image and returns as output the translation of that text into English. However if we take a look inside that box we can see two different systems, a Recognition system that recognizes the input text image and returns a recognized text, and a Translation system that translates the recognized text into English.

Many examples can be mentioned for applying the text image translation system in our daily life, for example, translation of text written on street posters, walls or paper to English that in some sense could help tourists to communicate with people who use a different language. Another example could be the transcription and translation of scanned documents which could have special interest for researchers especially when searching for information in different languages. This can be seen as a search engine similar to those currently existing for the web. It is at this point that the benefits of using HTR and MT systems are appreciated, since they could be used to automatically transcribe and translate documents drastically reducing the required translation time.

Late researches have shown much interest in the recognition and translation of printed and handwritten Arabic. It presents unique challenges and benefits and has been recently approached more than other scripts. Arabic is spoken by more than 234 million people and important in the cultures of many more [1]. It is one of the six United Nations official languages [2, 3, 4, 5]. The characters of Arabic script and similar characters are used by a high percentage of the world's population to write languages such as Arabic, Farsi (Persian), and Urdu. Arabic script differs from Latin scripts in several ways.

Unlike English handwriting, Arabic is written from right-to-left and does not distinct between upper or lowercase characters [6].

Although there are a few existing systems to translate common languages like German and Spanish text images into English, there are less systems to translate Arabic text images into English. In 2009, *Yi Chang et al* [7] reported an image-based automatic Arabic translation system that applies text detection based on GBT, SVM, and location-based prior knowledge. For the OCR module they use commercial software, the models of which are trained in a specified domain, to recognize the Arabic text. For the translation module they use the CMU PanDoRA translation system.

The most recent application that has reported acceptable results in this area is Google Googles for Mobile devices. The application captures an image containing text, detects the text automatically, and recognizes and translated the text to the desired language. However this application does not support Arabic recognition.

The main objective of this work is to develop a system that is able to recognize the Arabic language text in document images and translate the recognized text into accurate and fluent English. That is, computer systems that are able to emulate the human ability to read Arabic and translate it into English. In the case of handwritten recognition of text images, our work has focused on the use of the *embedded Bernoulli (mixture) HMMs (BHMMs),* that is, embedded HMMs in which the emission probabilities are modeled with Bernoulli mixtures [8]. These models have shown improvements among other systems in different corpus for Arabic Text [9, 10] In the case of Arabic text translation, we discuss three different state-of-the-art translation models: the Standard phrase-based model, the Hierarchical phrase-based model, and the Bilingual N-gram models. However our work has only focused on the phrase-based log-linear translation models, In particular, we used Moses [11] toolkit to carry out our experiments. The objective of this work can be divided into three major steps:

- Developing a Handwritten Arabic recognition system based on *embedded Bernoulli (mixture) HMMs (BHMMs).*

- Developing a Statistical Machine Translation System (SMT) based on the state-of-the-art log-linear models, in particular, using the Moses toolkit.

- Merging both system mentioned above in only one system that takes a new text image containing Arabic text as an input and generates the translation of that text into English as an output.

All results obtained in this work have been submitted to the NIST OpenHaRT 2013 evaluation [12] and very competitive results were achieved. Finally, the idea of an *Arabic Transcription and Translation system* for mobile applications is presented and discussed.

In what follows, we review the transcription (Chapter. 2), and translation systems (Chapter. 3). After that, we outline experiments and their results, as well as the employed tools in Chapter 4. Finally, we discuss the idea of implementing a mobile application in Chapter. 5

Chapter 2

Transcription System

In this work, we have applied the Windowed Bernoulli Hidden Markov models (BHMMs) with the use of vertical repositioning in order to recognize a new text image. We first apply a feature extraction process on that image which is converted into a sequence of feature vectors which are feed into the recognizer module. Finally, the recognition module obtains the most suitable transcription for the image. This recognition module is usually implemented as an statistical recognizer which has been automatically trained from labeled data. Below we describe the Bernoulli HMM with details.

2.1 Bernoulli HMM

Let $O = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$ be a sequence of feature vectors. An HMM is a probability (density) function of the form:

$$P(O \mid \mathbf{\Theta}) = \sum_{q_0,\ldots,q_{T+1}} \prod_{t=0}^{T} a_{q_t q_{t+1}} \prod_{t=1}^{T} b_{q_t}(o_t)\,, \tag{2.1}$$

where the sum is over all possible *paths* (state sequences) $q_0, \ldots, q_{T+1}$, such that $q_0 = I$ (special *initial* or *start* state), $q_{T+1} = F$ (special *final* or *stop* state), and $q_1, \ldots, q_T \in \{1, \ldots, M\}$, being M the number of regular (non-special) states of the HMM. On the other hand, for any regular states i and j, a_{ij} denotes the *transition* probability from i to j, while b_j is the *observation* probability (density) function at j.

A Bernoulli (mixture) HMM (BHMM) is an HMM in which the probability of observing $\mathbf{o}_t$, when $q_t = j$, is given by a Bernoulli mixture probability function for the state j:

$$b_j(\mathbf{o}_t) = \sum_{k=1}^{K} \pi_{jk} \prod_{d=1}^{D} p_{jkd}^{o_{td}} (1 - p_{jkd})^{1-o_{td}}\,, \tag{2.2}$$

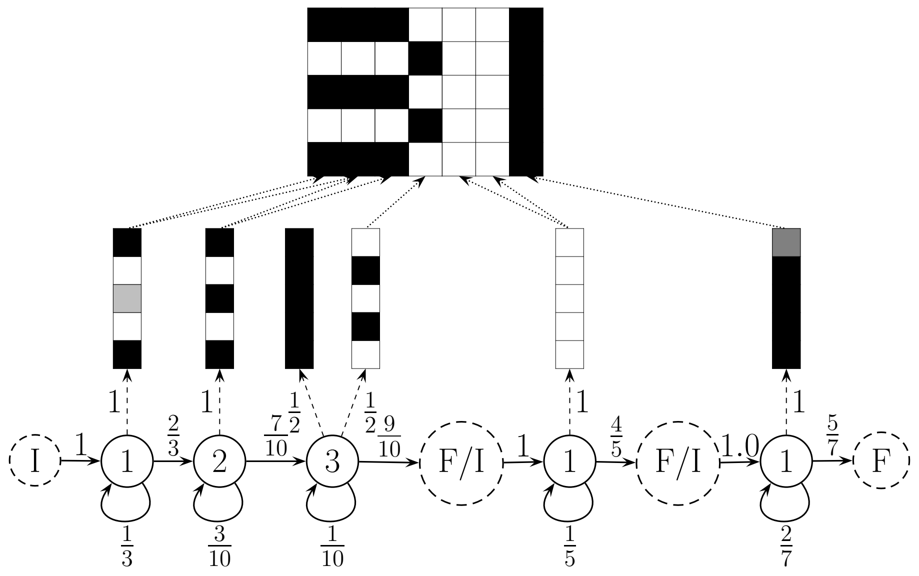


Figure 2.1: BHMM example for the number 31, together with binary image generated from it. Bernoulli prototype probabilities are represented using the following color scheme: black=1, white=0,gray=0.5 and light gray=0.1.

where π_{jk} and $\mathbf{p}_{jk}$ are, respectively, the prior and prototype of the kth mixture component in state j.

Bernoulli HMMs (BHMMs) at global (line or word) level are built from shared, embedded BHMMs at character level. More precisely, let C be the number of different characters (symbols) from which global BHMMs are built, and assume that each character c is modeled with a different BHMM of parameter vector $\mathbf{\Theta}_c$. Let $\mathbf{\Theta} = \{\mathbf{\Theta}_1, \ldots, \mathbf{\Theta}_C\}$, and let $O = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$ be a sequence of feature vectors generated from a sequence of symbols $S = (s_1, \ldots, s_L)$, with $L \leq T$. The probability of O can be calculated, using embedded HMMs for its symbols, as:

$$P(O \mid S, \mathbf{\Theta}) = \sum_{i_1,\ldots,i_{L+1}} \prod_{l=1}^{L} P(\mathbf{o}_{i_l}, \ldots, \mathbf{o}_{i_{l+1}-1} \mid \mathbf{\Theta}_{s_l}) \,, \tag{2.3}$$

where the sum is carried out over all possible segmentations of O into L segments, that is, all sequences of indices $i_1, \ldots, i_{L+1}$ such that

$$1 = i_1 < \cdots < i_L < i_{L+1} = T + 1;$$

and $P(\mathbf{o}_{i_l}, \ldots, \mathbf{o}_{i_{l+1}-1} \mid \mathbf{\Theta}_{s_l})$ refers to the probability (density) of the lth segment, as given by (2.1) using the HMM associated with symbol s_l.

Now, consider the Figure 2.1. An embedded BHMM for the number 31 is shown, which is the result of concatenating BHMMs for the digit 3, blank space and digit 1, in that order. Note that the BHMMs for blank space and digit 1 are simpler than that for digit 3. Also note that the BHMM for digit 3 is shared between the two embedded BHMMs shown in the Figure. The binary image of the number 31 shown above can only be generated from two paths, as indicated by the arrows connecting prototypes to image columns, which only differ in the state generating the second image column (either state 1 or 2 of the BHMM for the first symbol). It is straightforward to check that, according to (2.3), the probability of generating this image is 0.0004.

2.2 BHMM-based Handwriting Recognition

Given an observation sequence $O = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$, its most probable transcription is obtained by application of the conventional Bayes decision rule:

$$\begin{aligned} w^* &= \operatorname*{argmax}_{w \in W} \; p(w \mid O) && (2.4) \\ &= \operatorname*{argmax}_{w \in W} \; p(w)\, p(O \mid w)\,, && (2.5) \end{aligned}$$

where W is the set of possible transcriptions; $p(w)$ is usually approximated by an *n-gram language model* [13]; and $p(O \mid w)$ is a *text image model* which, in this work, is modeled as a BHMM (built from shared, embedded BHMMs at character level), as defined in Eq. (2.3). A particularly interesting case arises when the set of possible transcriptions reduces to a (small) finite set of *words (class labels).* In this case, $p(w)$ is simply the *prior* probability of word w, while $p(O \mid w)$ is the probability of observing O given that it corresponds to a handwritten version of word w.

2.2.1 The Viterbi algorithm

In order to efficiently compute $p(O \mid w)$ as a BHMM probability of the form given in Eq. (2.3), we use a dynamic programming methods known as *forward and backward algorithms* [14, 15]. Although these algorithms efficiently compute the exact value of $P(O \mid S, \mathbf{\Theta})$, it is common practice to approximate it by the so-called *Viterbi* or *maximum approximation,* in which the sums in Eqs. (2.1) and (2.3) are replaced by the max operator, i.e.

$$P(O \mid S, \mathbf{\Theta}) \approx \max_{\substack{i_1, \ldots, i_{L+1} \\ q_1, \ldots, q_T}} \prod_{l=1}^{L} \hat{P}(\mathbf{o}_{i_l}^{i_{l+1}-1} \mid \mathbf{\Theta}_{s_l})\,, \tag{2.6}$$

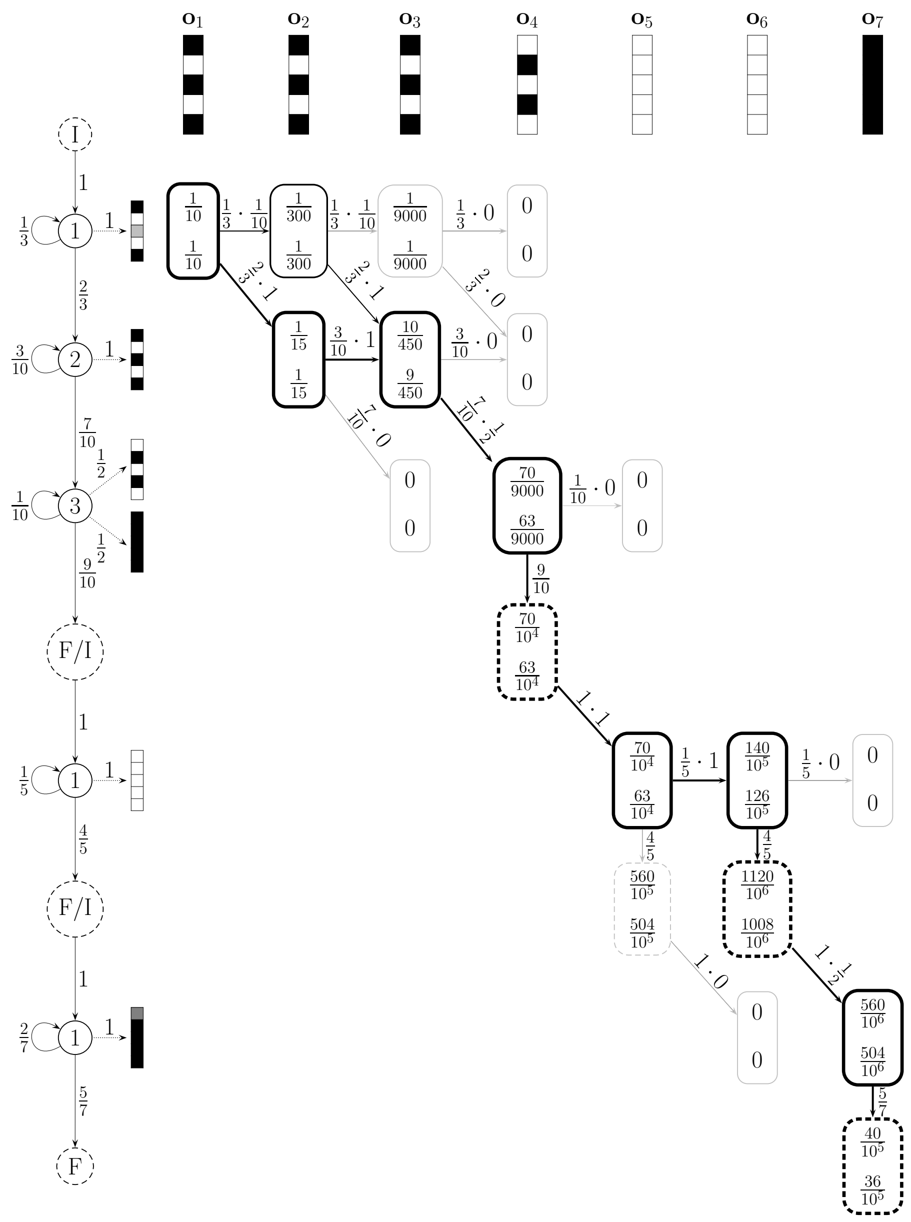


Figure 2.2: Application example of the forward and Viterbi algorithms to the the BHMM and observation of Figure 2.1 (bottom). Numbers at the top of the nodes denote forward probabilities, while those at the bottom refer to Viterbi scores.

where the $\hat{P}$ is defined as:

$$\hat{P}(\mathbf{o}_{i_l}^{i_{l+1}-1} \mid \mathbf{\Theta}_{s_l}) = a_{s_l I_{s_l} q_{i_l}} \cdot \prod_{t=i_l}^{i_{l+1}-2} a_{s_l q_t q_{t+1}} \cdot a_{s_l q_{i_{l+1}-1} F_{s_l}} \cdot \prod_{t=i_l}^{i_{l+1}-1} b_{s_l q_t}(\mathbf{o}_t)\,. \tag{2.7}$$

In contrast to the exact definition, this approximation allows us to identify a single, best state sequence or *path* associated with the given observation sequence. The well-known *Viterbi algorithm* efficiently computes this approximation, using dynamic programming recurrences similar to those used by the forward algorithm. Formally, we need to compute the probability $Q(l, t, j)$ of the most likely path up to time t that ends with the state j from the BHMM for symbol s_l. For the specials states, it can be computed as:

$$Q(l, t, I_{s_l}) = Q(l-1, t, F_{s_{l-1}}) \qquad \begin{matrix} 1 < l \le L \\ 1 \le t \le T \end{matrix}\;. \tag{2.8}$$

$$Q(l, t, F_{s_l}) = \max_{1 \le j \le M_{s_l}} Q(l, t, j)\, a_{s_l j F_{s_l}} \qquad \begin{matrix} 1 \le l \le L \\ 1 \le t \le T \end{matrix}\;, \tag{2.9}$$

while, for the regular states with $1 \le l \le L$ and $1 < t \le T$, we have:

$$Q(l, t, j) = \left[\max_{i \in \{I_{s_l}, 1, \ldots, M_{s_l}\}} Q(l, t-1, i)\, a_{s_l i j} \right] b_{s_l j}(\mathbf{o}_t)\,, \tag{2.10}$$

The base case is for $t = 1$:

$$Q(l, 1, i) = \begin{cases} a_{s_1 I_{s_1} i}\; b_{s_1 i}(\mathbf{o}_1) & l = 1, 1 \le i \le M_{s_1} \\ 0 & \text{otherwise} \end{cases}\;. \tag{2.11}$$

The Viterbi algorithm can be seen as a minor modification of the forward algorithm in which only the most probable is considered in each node computation. Indeed, the application example shown in Figure 2.2 is used both, for the forward and Viterbi algorithms. Now, however, the relevant numbers are those included at the bottom of each node, which denote $Q(l, t, j)$; i.e., at row 2 and column 3, we have $Q(1, 3, 2) = \frac{9}{450}$. Consider the generation of the third observation vector at the second state (for the first symbol). It occurs after the generation of the second observation vector, either at the first or the second states, but we only take into account the most likely case. Formally, the corresponding Viterbi score is computed as:

$$Q(1, 3, 2) = \max\left\{ \frac{1}{15} \cdot \frac{3}{10} \cdot 1, \frac{1}{300} \cdot \frac{2}{3} \cdot 1 \right\} = \max\left\{ \frac{9}{450}, \frac{1}{450} \right\} = \frac{9}{450}$$

Note that forward probabilities do not differ from Viterbi scores up to $Q(1, 3, 2)$, since it corresponds to the first (and only) node with two incoming paths.

The Viterbi approximation to the exact probability of generating the observation sequence is obtained at the final node: $Q(3,7,F) = 0.00036$. The most likely path, drawn with thick lines, is retrieved by starting at this node and moving backwards in time in accordance with computation of Viterbi scores. As usual in practice, the final Viterbi score in this example (0.00036) is a tight lower bound of the exact probability (0.00040).

2.3 Maximum likelihood parameter estimation

Maximum likelihood estimation of the parameters governing an embedded BHMM does not differ significantly from the conventional Gaussian case, and it can be carried out using the well-known EM (Baum-Welch) re-estimation formula [14, 15]. Let $(O_1, S_1), \ldots, (O_N, S_N)$, be a collection of N training samples in which the nth observation has length T_n, $O_n = (\mathbf{o}_{n1}, \ldots, \mathbf{o}_{nT_n})$, and was generated from a sequence of L_n symbols ($L_n \leq T_n$), $S_n = (s_{n1}, \ldots, s_{nL_n})$. At iteration r, the E step requires the computation, for each training sample n, of their corresponding forward and backward probabilities , as well as the expected value for its tth feature vector to be generated from kth component of the state j in the HMM for symbol s_l,

$$z_{nltk}^{(r)}(j) = \frac{\pi_{s_{nl}jk}^{(r)} \prod_{d=1}^{D} {p_{s_{nl}jkd}^{(r)}}^{o_{ntd}} (1 - p_{s_{nl}jkd}^{(r)})^{1-o_{ntd}}}{b_{s_{nl}j}^{(r)}(\mathbf{o}_{nt})},$$

for each t, k, j and l.

In the M step, the Bernoulli prototype corresponding to the kth component of the state j in the HMM for character c has to be updated as:

$$\mathbf{p}_{cjk}^{(r+1)} = \frac{1}{\gamma_{ck}(j)} \sum_{n} \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi_{nltk}^{(r)}(j) \mathbf{o}_{nt}}{P(O_n \mid S_n, \boldsymbol{\Theta}^{(r)})}, \tag{2.12}$$

where $\gamma_{ck}(j)$ is a normalization factor,

$$\gamma_{ck}(j) = \sum_{n} \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi_{nltk}^{(r)}(j)}{P(O_n \mid S_n, \boldsymbol{\Theta}^{(r)})}, \tag{2.13}$$

and $\xi_{nltk}^{(r)}(j)$ the probability for the tth feature vector of the nth sample, to be generated from the kth component of the state j in the HMM for symbol s_l,

$$\xi_{nltk}^{(r)}(j) = \alpha_{nlt}^{(r)}(j) z_{nltk}^{(r)}(j) \beta_{nlt}^{(r)}(j). \tag{2.14}$$

Similarly, the kth component coefficient of the state j in the HMM for character c has to be updated as:

$$\pi_{cjk}^{(r+1)} = \frac{1}{\gamma_c(j)} \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi_{nltk}^{(r)}(j)}{P(O_n \mid S_n, \mathbf{\Theta}^{(r)})}\,, \tag{2.15}$$

where $\gamma_c(j)$ is a normalization factor,

$$\gamma_c(j) = \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \alpha_{nlt}^{(r)}(j)\beta_{nlt}^{(r)}(j)}{P(O_n \mid S_n, \mathbf{\Theta}^{(r)})}\,. \tag{2.16}$$

To avoid null probabilities in Bernoulli prototypes, they can be smoothed by linear interpolation with a flat (uniform) prototype, $\mathbf{0.5}$,

$$\tilde{\mathbf{p}} = (1-\delta)\,\mathbf{p} + \delta\,\mathbf{0.5}\,, \tag{2.17}$$

where, for instance, $\delta = 10^{-6}$.

2.4 Windowed BHMMs

Given a binary image normalized in height to H pixels, we may think of a feature vector $\mathbf{o}_t$ as its column at position t or, more generally, as a concatenation of columns in a window of W columns in width, centered at position t. This generalization has no effect neither on the definition of BHMM nor on its maximum likelihood estimation, though it might be very helpful to better capture image context at each horizontal position of the image. As an example, Figure 2.3 shows a binary image of 4 columns and 5 rows, which is transformed into a sequence of 4 15-dimensional feature vectors (first row) by application of a sliding window of width 3. For clarity, feature vectors are depicted as 3×5 subimages instead of 15-dimensional column vectors. Note that feature vectors at positions 2 and 3 would be indistinguishable if, as in our previous approach, they were extracted with no context ($W = 1$).

Although one-dimensional, "horizontal" HMMs for image modeling can properly capture non-linear horizontal image distortions, they are somewhat limited when dealing with vertical image distortions, and this limitation might be particularly strong in the case of feature vectors extracted with significant context. To overcome this limitation, we have considered three methods of window *repositioning* after window extraction: *vertical, horizontal,* and *both.* The basic idea is to first compute the compute the center of mass of the extracted window, which is then repositioned (translated) to align its center to the center of mass. This is done in accordance with the chosen method, that is, horizontally, vertically, or in both directions. Obviously, the feature vector actually extracted is that obtained after repositioning. An example of feature extraction is shown in Figure 2.3 in which the the standard method (no repositioning) is compared with the three methods repositioning methods considered.

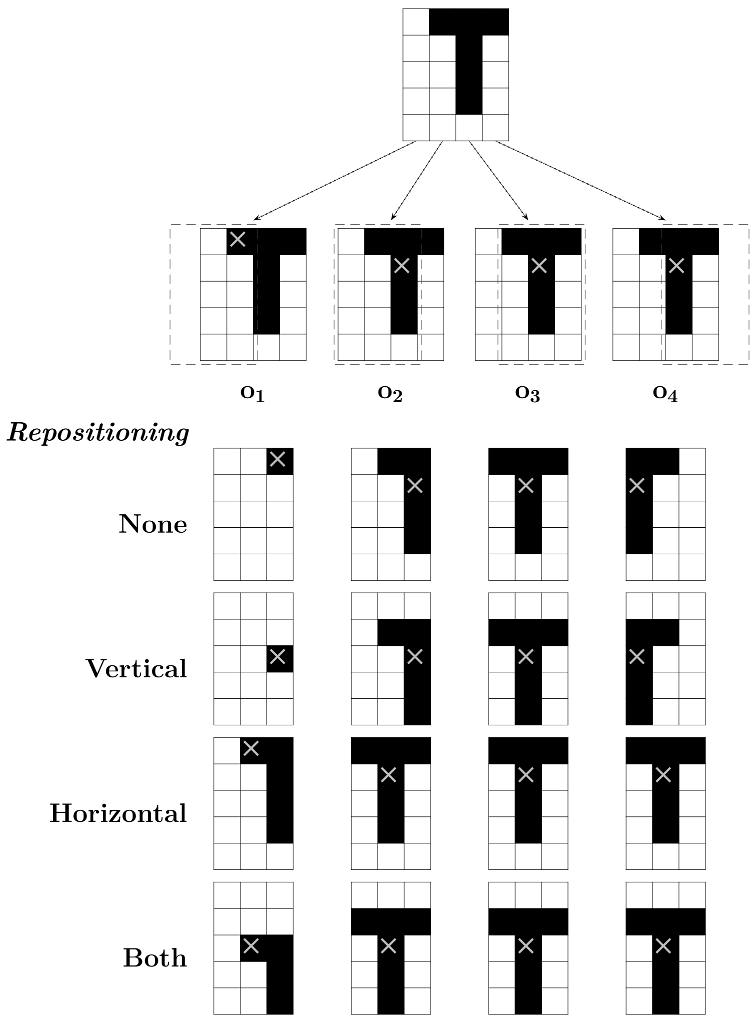


Figure 2.3: Example of transformation of a 4×5 binary image (top) into a sequence of 4 15-dimensional binary feature vectors $O = (\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3, \mathbf{o}_4)$ using a window of width 3. After window extraction (illustrated under the original image), the standard method (no repositioning) is compared with the three repositioning methods considered: vertical, horizontal, and both directions. Mass centers of extracted windows are also indicated.

Chapter 3

Translation System

In this chapter we review the state-of-art applications and approaches that we used to carry out our experiments in the field of *Statistical Machine Translation (SMT)*. We might think of a SMT as a task to automatically translate a source sentence x into a target sentence y. The system is to select the sentence with the higher probability among all possible y.

$$x = x_1 ... \, x_j, x_j \in \mathbf{X}, \; j = 1, ..., J \tag{3.1}$$

$$y = y_1 ... \, y_i, x_i \in \mathbf{Y}, \; i = 1, ..., I \tag{3.2}$$

where x_j and y_i denote source and target words; and $\mathbf{X}$ and $\mathbf{Y}$, the source and target vocabularies respectively.

Nowadays, SMT systems follow the Bayes decision rule approach [16, 17] in which the optimal target sentence y is found by maximizing the posterior probability,

$$y^* = \underset{y}{\operatorname{argmax}} \; p(y \mid x) \tag{3.3}$$

$$= \underset{y}{\operatorname{argmax}} \; p(y) \, p(x \mid y) \, , \tag{3.4}$$

Applying Bayes' theorem we can re-write the first equation as shown above in (eq 3.4), where $p(y \mid x)$ is the *translation model*, and $p(y)$ is the *language model*. The language model describes the correctness of the target language sentence which helps to avoid syntactically incorrect sentences. The translation model is decomposed into *lexicon model* and *alignment model*.

Nevertheless, most of the current statistical MT systems present an alternative modeling of the translation process different from that presented in Eq. 3.4. The posterior probability is modeled as a log-linear combination of feature functions [18] as follow

$$y^* = \underset{y}{\operatorname{argmax}} \sum_{m=1}^{M} \lambda_m h_m(x, y), \tag{3.5}$$

with λ_m being the log-linear interpolation weight and $h_m(x, y)$ is a feature function, such as the logarithm of a language model, or the logarithm of a phrased-based model.

3.1 Word Alignment Model

In this section, we will describe how to obtain some of the model parameters. The most important one is the phrase probability translation table that maps foreign phrases (Arabic) to English phrases. One of the most common tool to establish a word alignment is to use the toolkit *GIZA++*. This toolkit is an implementation of the *IBM Models*. IBM Model 1 uses only lexical translation probabilities, Model 2 adds an absolute alignment model, Model 3 adds a fertility model, Model 4 replaces the absolute alignment model with a relative alignment model, and Model 5 fixes a problem with deficiency in the model.

These models have some serious draw-backs. Most importantly, they only allow at most one English word to be aligned with each foreign word. To resolve this, some transformations are applied: First, the parallel corpus is aligned bidirectionally, Arabic to English and English to Arabic. This generates two word alignments that have to be reconciled. If we intersect the two alignments, we get a high-precision alignment of high-confidence alignment points. If we take the union of the two alignments, we get a high-recall alignment with additional alignment points. See figure 3.1 for an illustration.

3.2 Phrase-based Models

Phrase-based statistical machine translation models are based on the translation of phrases instead of words as atomic units. A phrase can be defined as a continuous multiword sequence. Phrases are mapped one-to-one based on a phrase translation table, and may be reordered. A Phrase translation table can be learned based on a word alignment. Below we explain three different phrase-based models:

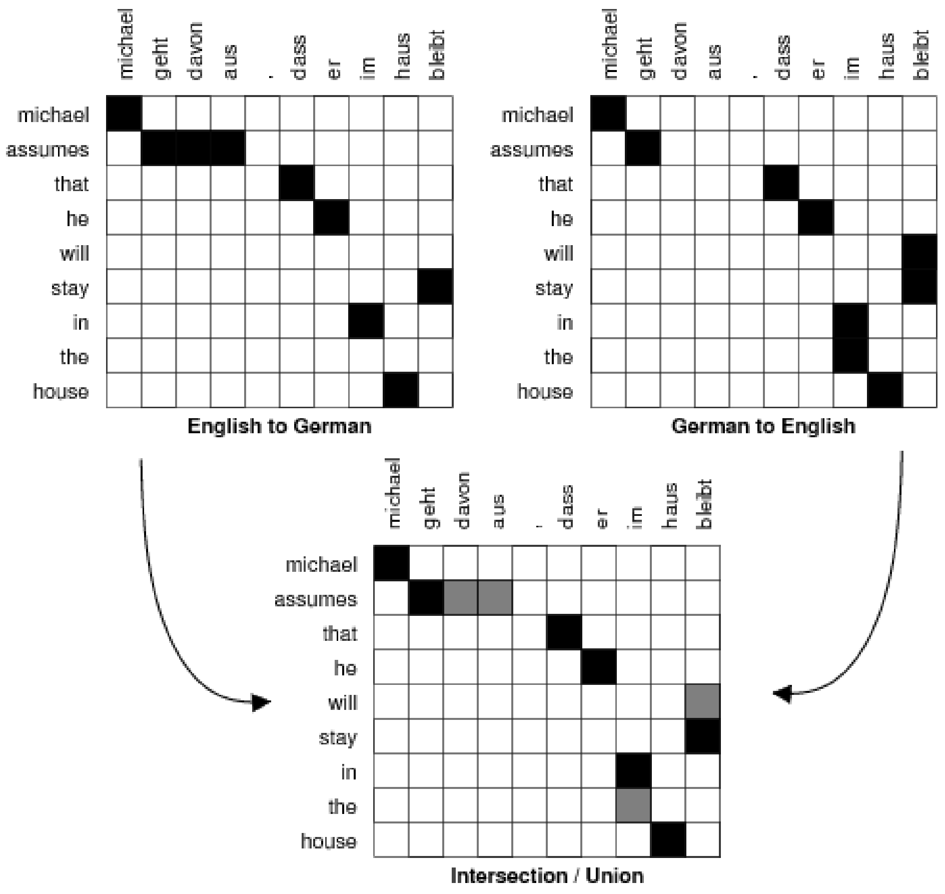


Figure 3.1: Merging alignments by taking the intersection (black) or union(gray) of sets of alignment points

3.2.1 Standard phrase-based models

The heuristic estimation of phrase-based models is grounded on the Viterbi alignments computed as a byproduct of word-based alignment models. The Viterbi alignment is defined as the most probable alignment given the source and target sentences and an estimation of the model parameters. An good example of this type is the **Moses** toolkit [11]. Moses is a complete out-of-the-box translation system for academic research. It consists of all the components needed to preprocess data, train the language models and the translation models. It also contains tools for tuning these models using minimum error rate training and evaluating the resulting translations using the BLEU score. Moses uses GIZA++ for word alignments and SRILM for language modeling which are standard external tools.

3.2.2 Hierarchical phrase-based models

They are one of the current promising approaches to SMT. They take into account a weighted synchronous context-free grammar is induced from parallel text. In addition to contiguous lexical phrases, hierarchical phrases with usually up to two gaps are extracted. Hierarchical decoding is carried out with a search procedure which is based on CYK+ parsing (Chappelier and Rajman, 1998). A good example on these models is **Jane** toolkit [19]. Jane is an open source translation toolkit which has been developed at RWTH Aachen University and is freely available for non-commercial use. Jane provides efficient C++ implementations for hierarchical phrase extraction, optimization of log-linear feature weights, and parsing-based search algorithms.

3.2.3 Bilingual Ngram models approach

This approach can be seen as an alternative to the standard phrase-based approach [20, 21]. A good example on this approach is the **Ncode** toolkit [22]. Ncode is an open source statistical machine translation decoder and its companion tools. Ncode main features include the use of multiple n-gram language models estimated over bilingual units, source words and/or target words or any factor decomposition, lexicalized reordering, several tuple (unigram) models, etc.. As for nearly all current statistical approaches to machine translation, these models are embedded in a linear model combination. Ncode splits the reordering and decoding problems of SMT in two separate modules, aiming at better tackling each of the problems. However, hard reordering decisions are avoided by means of using permutation lattices.

Chapter 4

Experiments

Experiments in this work were carried out on the Arabic-English parallel corpus provided by the Linguistic Data Consortium (LDC) on the NIST Open Handwriting Recognition and Translation evaluation (OpenHaRT). The data reported here is collected from both 2010 and 2013 evaluations. Both evaluations focus on core recognition and translation technologies for document images containing primary Arabic handwritten scripts [23, 12]. The focus of the first evaluation was not only centered on the recognition and translation of Arabic text but also on word and line segmentation which was represented as a series of polygon coordinates indicating the locations of the text segments within the image. In the last evaluation, the main focus is centered on recognition and translation of Arabic text so as our main approach.

4.1 NIST OpenHaRT corpus

The National Institute of Standards and Technology (NIST) Open Handwriting Recognition and Translation (OpenHaRT) evaluation is a public evaluation of image-to-text transcription and translation, similar to the tasks evaluated by NIST for the DARPA Multilingual Automatic Document Classification Analysis and Translation (MADCAT) Program, see [23, 12]. The 2010 and 2013 evaluations focus on recognition and translation of images containing primary Arabic handwritten script.

Data for this evaluation was created by the Linguistic Data Consortium (LDC) and has been used in previous MADCAT evaluations. This data was created in a controlled environment where known scribes copied Arabic source texts that were previously used in the DARPA GALE program. The source text was originally in electronic format. A corresponding document image was created by instructing literate native Arabic writers to produce handwritten copies of chosen passages using various writing conditions. Each passage was copied by at least two scribes. The handwritten copies were then scanned at 600 dpi to create the document images in TIFF format. A writing factor is considered, the writing instrument, surface and speed. Please refer to the NIST evaluation plan for details [23, 12].

In this work, we will only focus on the 2013 NIST OpenHaRT evaluation [12]. Thus, we will take into account only the segmentation conditions and training conditions considered in this evaluation. The OpenHaRT 2013 database consists of 43590 Arabic documents for training, 1004 Arabic documents for development, and 633 Arabic documents for testing. More details about this corpus is shown in Table 4.1 for unique writer.

Dataset	All Documents	Segments	Words
Training	43590	41584	900121
Development	1004	916	20099
Testing	633	3144	69393

Table 4.1: NIST OpenHaRT 2013 database statistics extracting words and segments for a unique writer

The 2010 evaluation process was paired with two segmentation conditions to explore the relationship between system performance and the system's ability to segment the data. Segmentation is represented as a series of polygon coordinates indicating the locations of the text segments within the image. The two segmentation conditions are referred to as word segmentation and line segmentation. On the other hand, the 2013 evaluation process was only paired with line segmentation condition. Below is a brief description about each condition.

- **Word segmentation** is created manually. Human annotators mark the word boundaries using the GEDI tool. The input to GEDI is a document image.

- **Line segmentation** is the primary segmentation condition. It is defined as a bounding box that surrounds a line of text and is derived algorithmically from the word segmentations by creating polygons that minimize the amount of text overlap between the lines.

4.2 Preprocessing and Training

4.2.1 Transcription System

In this work, we have applied the windowed BHMMs (Bernoulli HMMs) [24, 8, 25]. Each transcription hypothesis is built from an HMM in which emission probabilities are modelled as Bernoulli mixture distributions. To keep the number of independent parameters low, the BHMM at sentence level (transcription hypothesis) is built from BHMMs at character level which depend on their surrounding characters, the so-called *tri-character modelling approach*. Given a text image of an unknown word, each windowed BHMM computes the probability of the given image to be a handwritten version of its corresponding word. To compute these probabilities, text images are first transformed into a sequence of binary feature vectors by applying a sliding window at each horizontal position. The width of the sliding window is known to have a strong effect on the system ability to capture local image distortions, and thus this parameter has to be tuned. Moreover, we have recently observed that local image distortions, and vertical distortions in particular, might not be properly modeled when the sliding window is applied at a constant vertical position of the image. To overcome this limitation, we applied *repositioning* on the sliding window before its actual application. That is, the sliding window was repositioned so as to align its center with its mass center. In this work, we applied only a vertical repositioning due to its better performance over another two methods (horizontal and in both directions) discussed in [9, 26, 27].

Keeping the tri-character approach in mind, a list of tri-characters was obtained by taking the first $N \in \{50, 100, 200, 500\}$ frequent ones, that is, if a tri-character T appears more than N time, it will be selected. Selected ones were replaced with those of uni-characters to avoid duplication. This approach improved our results since we model for the character and it's surrounding two characters. We used $N = 500$ to do our experiments since it was the best in results, number of characters, and time consuming.

The transcription system was trained from input images scaled in height to 30 pixels (while keeping the aspect ratio), and then binarized with the Otsu algorithm [28]. A sliding window of width 9 using the vertical repositioning was applied, and thus the resulting input (binary) feature vectors for the BHMMs had 270 bits. Since in Arabic, the shape of a letter written at the beginning of the word is different from a letter written at the middle or at the end; all Arabic transcriptions were encoded by adding this shape information.

Finally, the number of states per character was adjusted to 6 states for all BHMMs. Similarly, the number of mixture components per state was empirically adjusted to 128. Parameter estimation and recognition were carried out using the EM algorithm. Also, we used a 5-gram language model at character level instead of the conventional class priors. The language model was smoothed by linear interpolated estimates with absolute modified Kneser-Ney discounting. In addition, the grammar scale factor was adjusted to 30.

4.2.2 Translation System

As mentioned in the introduction, the translation system for the translation task is based on one of the state-of-the-art machine translation systems. Each systems use a different techniques to perform a good translation. In this work we discussed only three of them: Standard phrase-based models, Hierarchical phrase-based models, Bilingual Ngram models approach (Sec. 3.2). In this work, we had have to choose only one of them to perform the translations of our system. After comparing between them on the Arabic-English corpus introduced by NIST OpenHaRT 2013 [12], we found that the log-linear translation system, Moses [11], has performed the best translations.

Specifically, in our system, we used the standard Moses features: a phrased-based model that includes both direct and inverse phrase translation probabilities and both direct and inverse lexical weights, a language model, a distance-based reordering model, a word penalty, and a lexicalized reordering model. In the case of the language model, we used a *5-gram* model trained with SRILM [29]. This model was smoothed by linear interpolated estimates with absolute modified Kneser-Ney discounting.

Each source and target sentence was pre-processed. English text was tokenized with Moses tokenization tools [11], and Arabic text was tokenized using the *MADA+TOKAN* tool [30]. Additionally, long sentences (longer than 150 words) were then removed. Finally, standard Moses training was performed on the training data, which includes: alignment extraction, phrase extraction and MERT [11].

4.3 Experiments and Submissions

Transcription and Translation systems are typically based on the concatenation of two systems: a Handwritten Text Recognition (HTR) system and a Machine Translation (MT) system. In this section, we describe and test each system alone and in conjunction, which can be classified into three different tasks: Document Image Recognition (DIR) task, the Document Text Translation (DTT) task, and the Document Image Translation (DIT) task.

The name of these tasks was used in the NIST OpenHaRT 2010 and 2013 evaluations. All experiments of this section, was submitted to the NIST 2013 evaluation. Below, our participation is discussed in details.

For the 2013 NIST OpenHaRT 2013 evaluation, many systems were submitted for three different tasks, the Document Image Recognition (DIR) task, the Document Text Translation (DTT) task, and the Document Image Translation (DIT) task. Systems were trained following two training conditions: a constrained condition that required participants to develop their systems using only the provided LDC data resources, and an unconstrained condition in which participants are free to use any additional publicly available non-LDC resources for the system development (For more information, please refer to [12]).

For the DIR task, the UPV submitted *two* systems (*DIR1*, the primary system, and *DIR2*, the contrastive system) that followed the constrained training condition. They used the BHMMs described in Sec. 2. The only difference between the systems is that the DIR2 system was trained using the complete data set, whereas the DIR1 system was trained using less data. Statistics about the data used to train both systems (DIR1 and DIR2) are reported in Table 4.2.

For the DTT, *two* primary systems were submitted. The first one followed the constrained training condition (DTT constrained), while the other one followed the unconstrained training condition (DTT unconstrained). Both systems were trained using the system described in Section. 3. However, for the unconstrained task, we used some of the freely available data that was used in *IWSLT 2011* challenge: *MultiUN* [31] and *TED* [32]. Since *MultiUN* corpus is not aligned at sentence level, we used the Champollion [33] tool for aligning the sentences. Finally, we selected sentences for the training set according to the infrequent n-grams score [34], in order to gather a specific training set to translate our source test sentences. It is worth noting that the number of sentences used for training was $20K$ from MultiUN and $2K$ from TED. Further statistics about each corpus used to train our translation systems in both conditions (DTT constrained and DTT unconstrained) are shown in Table 4.3. We used around $40K$ of data segments to train our system following the constrained condition. However, we used about $62K$ of data segments to train the DTT system following the unconstrained condition.

Given a handwritten image f, the DIT task, can be expressed as follows,

$$y^{\star} = \underset{y \in Y}{\operatorname{argmax}}\ p(y|f) = \underset{y \in Y}{\operatorname{argmax}}\ \sum_{x} p(x|f)\ p(y|x) \tag{4.1}$$

where x stands for a candidate recognized source (Arabic) text and y for a candidate translated sentence (in English) corresponding to the input image f.

Since the summation over all possible transcriptions in Eq. (4.1) cannot be computed in practice, for the Document Image Translation (DIT) task, we submitted three different systems. In all of them, the probability $p(x \mid f)$ in Eq. (4.1) was approximated by the primary DIR transcription system. Therefore, the key difference among systems lay in the translation subsystems.

In the primary DIT system (DIT1), Eq. (4.1) was approximated as follows,

$$\begin{aligned} y^* &\approx \underset{y \in Y}{\operatorname{argmax}}\ [\max_x \{p(x|f)\ p(y|x)\}] \\ &\approx \underset{y \in Y}{\operatorname{argmax}}\ [p(y| \max_x \{p(x|f)\})] \end{aligned} \tag{4.2}$$

and $p(y|x^\star)$ was approximated by the primary DTT translation system. In other words, the input image was recognized by the primary DIR transcription system, and the recognized text was fed into the primary DTT translation system.

The second DIT system (DIT2) followed a similar approach to that of the first DIT system, approximating Eq. (4.1) by Eq. (4.2). However, the translation probability was approximated by a translation system analogous to the primary DTT system but trained differently. In this case, the source part of each bilingual training pair was substituted by the transcription obtained by the primary DIR system. The new training data set produced in this way was used to train the translation system. This second translation system was expected to better handle the noisy output of the DIR system. Accordingly, this system showed a better performance than the standard (primary) system in the development set. However, in the test set it showed a worse performance. For further details, please refer to Table 4.5.

Finally, in the third system (DIT3), a different approximation of Eq. (4.1) was used

$$y^\star = \underset{x \in \mathrm{NBest}(f)}{\operatorname{argmax}} \left\{ \underset{y \in \mathrm{NBest}(f|x)}{\operatorname{argmax}} \{p(x|f)\ [p(y|x)]^\theta\} \right\} \tag{4.3}$$

where we introduced a scaling factor θ, and the search space was approximated by N-best lists. Specifically, each input image was first recognized using the primary DIR system into 100-Best transcriptions, and then each transcription was translated using the primary DTT system into 100-Best translations. Finally, the optimal scaling factor θ was found using a grid search in a development set so as to maximize the BLEU.

In Tables 4.2 and 4.3 (last row), we report the data used to train each part of our Recognition and Translation System in the constrained condition. For the recognition part, we used about $779K$ of data lines for training, and for the translation part we used around $40K$ of data segments for training.

Table 4.2: Data (lines) used for training each system and its training conditions.

System/Condition	Constrained	Unconstrained
DIR1	779, 100	-
DIR2	789, 874	-
DIT (recognition part)	779, 100	-

Table 4.3: Data (segments) used for training each system and its training conditions.

System/Condition	Constrained	Unconstrained	
Corpus	LDC	MultiUN	TED
DTT	40, 580	19, 956	2, 205
DIT (translation part)	40, 580	-	-

4.4 Results

In this section, we summarize the results obtained in the OpenHaRT 2013 evaluation for all presented systems. For recognition systems, results are shown in terms of Word Error Rate (WER%), whereas for translation systems, results are shown in terms of BLEU score. In Table 4.4, results for the two DIR systems (DIR1 and DIR2) are reported on the EVAL set [12] (Eval'13 column). Also, these systems, in particular DIR1, was compared with the OpenHaRT 2010 system (UPV PRHLT) for DIR and line segmentation condition. This comparison was performed by evaluating both systems on the DRYRUN set [12] (Eval'10 column). It is worth noting that the evaluation set in the OpenHaRT 2010 is the development set in the OpenHaRT 2013 (Eval'10 column). Having this in mind, we can easily compare our previous results obtained in OpenHaRT 2010 with results obtained in the DRYRUN set of the OpenHaRT 2013. On the other hand, Table 4.5 reports results of the DTT system for both training conditions (constrained DTT and unconstrained DTT) together with the three DIT systems (DIT1, DIT2, and DIT3). These systems were evaluated on both sets EVAL and DRYRUN (Eval'10 and Eval'13 columns respectively). The evaluation on EVAL set was performed by NIST. However, the evaluation on DRYRUN set was performed by UPV. The UPV evaluation procedure might has slightly differed from the NIST procedure.

Table 4.4: Submitted systems for DIR and line segmentation condition together with their Word Error Rate (WER%)

System	Reference	WER [%]	
		Eval'10	Eval'13
DIR1	p-1_1_20130425	29.08	29.32
DIR2	c-1_2_20130425	-	**29.20**
UPV PRHLT	OpenHaRT'10	47.45	-

As shown in Table 4.4, the DIR2 system slightly outperforms the DIR1 system. This conclusion was obviously expected for us since DIR2 system was trained with more data. Additionally, both DIR1 and DIR2 systems outperform our system (UPV PRHLT) from the OpenHaRT 2010 evaluation. This was also expected because in this evaluation we trained our models with more mixture components (128) per state, and also we used a bigger language model for recognition.

Table 4.5: Submitted systems for (DTT and DIT) and line segmentation condition together with their BLEU score

System	Reference	BLEU [%]	
		Eval'10	Eval'13
DTT Constrained	p-1_1_20130425	22.53	21.93
DTT Unconstrained	p-1_1_20130425	25.18	24.10
DIT1	p-1_1_20130425	16.51	16.95
DIT2	c-1_2_20130425	16.58	16.52
DIT3	c-1_3_20130425	18.13	**17.49**

As shown in Table 4.5, the usage of an additional small set of data (around $20K$) significantly improved the translation accuracy in the DTT system. More precisely, the Unconstrained DTT system significantly outperforms the Constrained DTT system. Here, we remind the reader that this additional data was selected according to the infrequent n-grams score [34], in order to gather a specific training set that relates to the source test sentences. In the same Table (4.5), the DIT3 shows better performance than DIT1 and DIT2. Specifically, in the DIT3 system, the search space was approximated by means of 100-best list. This approach helped in finding better transcriptions and translations which resulted in improving the results.

4.5 Tools and Means

In this section we describe the tools used in this work. For text pre-processing, we used the Moses tokenization tools [11] for English text tokenization. On the other hand, we used the MADA+TOKAN [30] toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In addition, we used the Champollion Toolkit (CTK) [33] to align the MultiUN [31] parallel corpus on sentence level.

For the handwritten text recognition system, we used the TLK [35] toolkit which among other features implements Bernoulli Hidden Markov models (BHMMs). This toolkit was developed by the UPV.

For the machine translation system, we used one of the state-of-the-art, phrase-based statistical machine translation systems, *Moses* [11]. To establish the word alignments of a parallel corpus, we used MGIZA++ [36].

For both handwritten text recognition and machine translation systems we used the SRI Language Modeling Toolkit (SRILM) [29] to generate the corresponding language models.

Chapter 5

System Design

5.1 Introduction

In this chapter, we introduce the idea of an Arabic Transcription and Translation service for mobile hand-held devices, including mobile telephones, Pocket PCs, and PDAs. This service, which is referred to *Image Translation*, is usually an additional service provided by mobile translation applications where the user can take a photo of some printed text (menu list, road sign, document etc.), apply optical character recognition (OCR) technology to it to extract any text contained in the image, and then have this text translated into a language of their choice. In this work, we show the process of translating Arabic image text to English only.

In order to support the machine transcription and translation service (Image Translation service), a mobile device needs to be able to communicate with external computers (servers) that receive the user-input image, recognize it into text, translate the recognized text and send the translated text back to the user. This is usually done via an Internet connection (WAP, GPRS, EDGE, UMTS, Wi-Fi) but some earlier applications used SMS to communicate with the translation server.

Some image translation applications also offer additional services that further facilitate the translated communication process, one example is the speech generation (speech synthesis), where the (translated) text may be transformed into human speech (by a computer that renders the voice of a native speaker of the target language)

Below we will discuss the possibility of implementing a live Arabic image translation service for Android mobiles. First of all, we will create a discuss a basic UML design and a database scheme. The database is used to store all images and texts from users (with their permission) for further investigation which can be used later to improve the quality of recognition and translation. It is worth noting that the more data we use to train our models (transcription and translation models), the more quality the output text will be. After that, we will discuss how the server would deal with the received data. Finally, we will show some photos of how this application works.

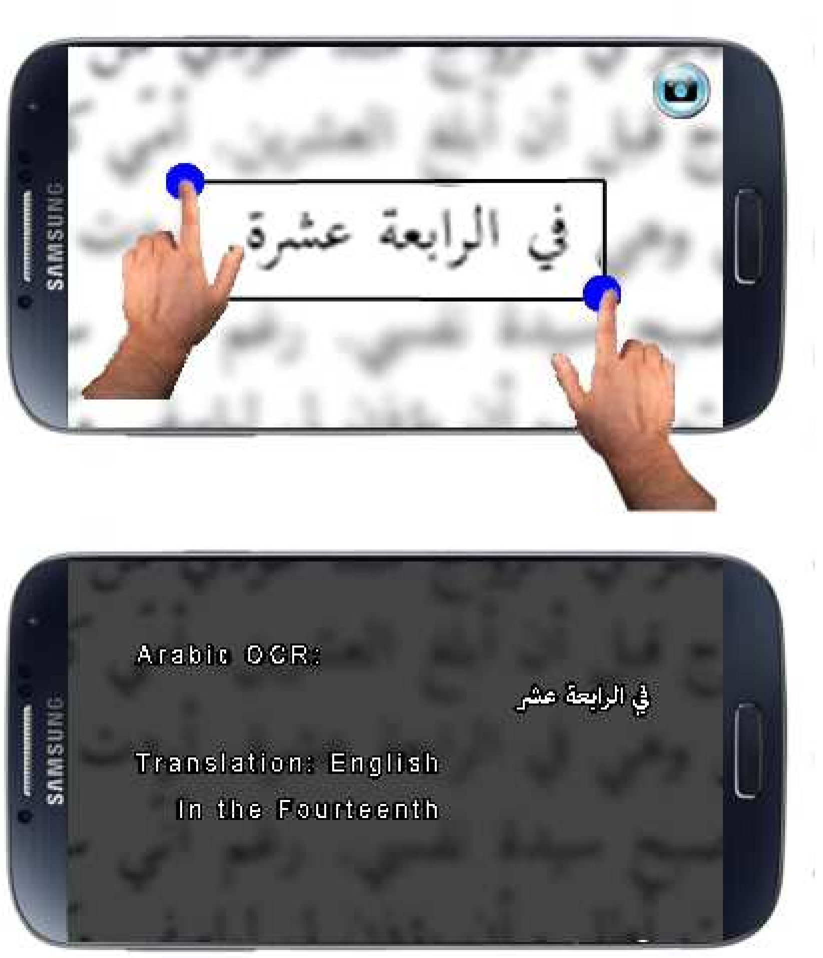
5.2 System Design and Mobile Application

The basic idea behind this kind of systems is to have a client-side application that communicates with a server-side application via sockets. That is, when a user pushes the camera button to capture an Arabic text image, the client-side application automatically sends the image via sockets to the server-side application which will process the image, recognize it, translate it, and finally send the results (translated text) back to the client-side application to be shown for the user. This complete process may vary in the speed depending on the Internet speed to send and receive the data. Also it depends on the time of processing in the server.

The client-side application is simple and easy to use. As soon as the user runs the application, the camera turns on. A shortcut to the capture button appears on the top right corner to permit a quick and accurate image capturing. Finally, in the middle of the screen, a dynamic box appears to place the required Arabic text to be translated inside it. Figure 5.1 shows the design of the Image translation mobile application. In the first image (top), the first screen of the application is shown. The dynamic box is bounding the text to be translated. When the user pushes the capture button on the top right corner of the screen, the application sends the data to the server to be translated. while this process is happening, a loading circle appears to let the user know that the request is being processed. When it is done, the results is shown as in Figure 5.1 second image (bottom). First the user should see the recognized text (Arabic OCR), then the translated text into English.

The server-side application is more complicated and it has more steps to do. First of all, we will define all components needed to do the server-side work with their memory usage, time consuming and model size on disk. Table 5.1 shows a list of components for each module (Feature Extraction, Recognition, and Translation) together with the memory usage, time consuming, and size on disk.

From Table 5.1, the feature extraction module is the module that consumes less time and memory among others (0.1 G of memory, and 1 second). On the other hand, the recognition module with its three models (Acoustic,



Arabic OCR:
في الرابعة عشر
Translation: English
In the Fourteenth

Figure 5.1: Image Translation system for Android Devices. First figure (top) shows the initial screen of the application with the Arabic text to be translated. Second figure (bottom) shows the results of translation

Component	Size(Mb)	Memory(Gb)	Time(s)
Feature Extraction		0.1	1
Recognition		3.0	125
-Language Model	300		
-Acoustic Model	1800		
-Lexicon Model	5		
Tokenizing		0.5	47
Translation		0.7	10
-Language Model	119		
-Phrase-based table	28		
-Reordering table	25		
Detokenizing		0.5	10
Content Storing		0.1	1
Total:	2277	4.8	193

Table 5.1: Server-side components with their memory usage and time consuming, and the models used with their size on disk

Language, and Lexicon models) consumes the most (3 Gb of memory and around 2 minutes). The Translation module has a pre-step to tokenize the data and a post-step to detokenize the results and to put them in the correct format. This module contains three models that takes around 10 seconds and consumes around 0.7 Gb of memory. The last module in this step is the Content Storing module. This module is responsible of storing some data about the request made by the user in the server. This module is disabled by default and it needs the permission of the user in order to work. In next section we explain how this module works, also we show the UML design of the database used to store the data.

5.3 Content Storing

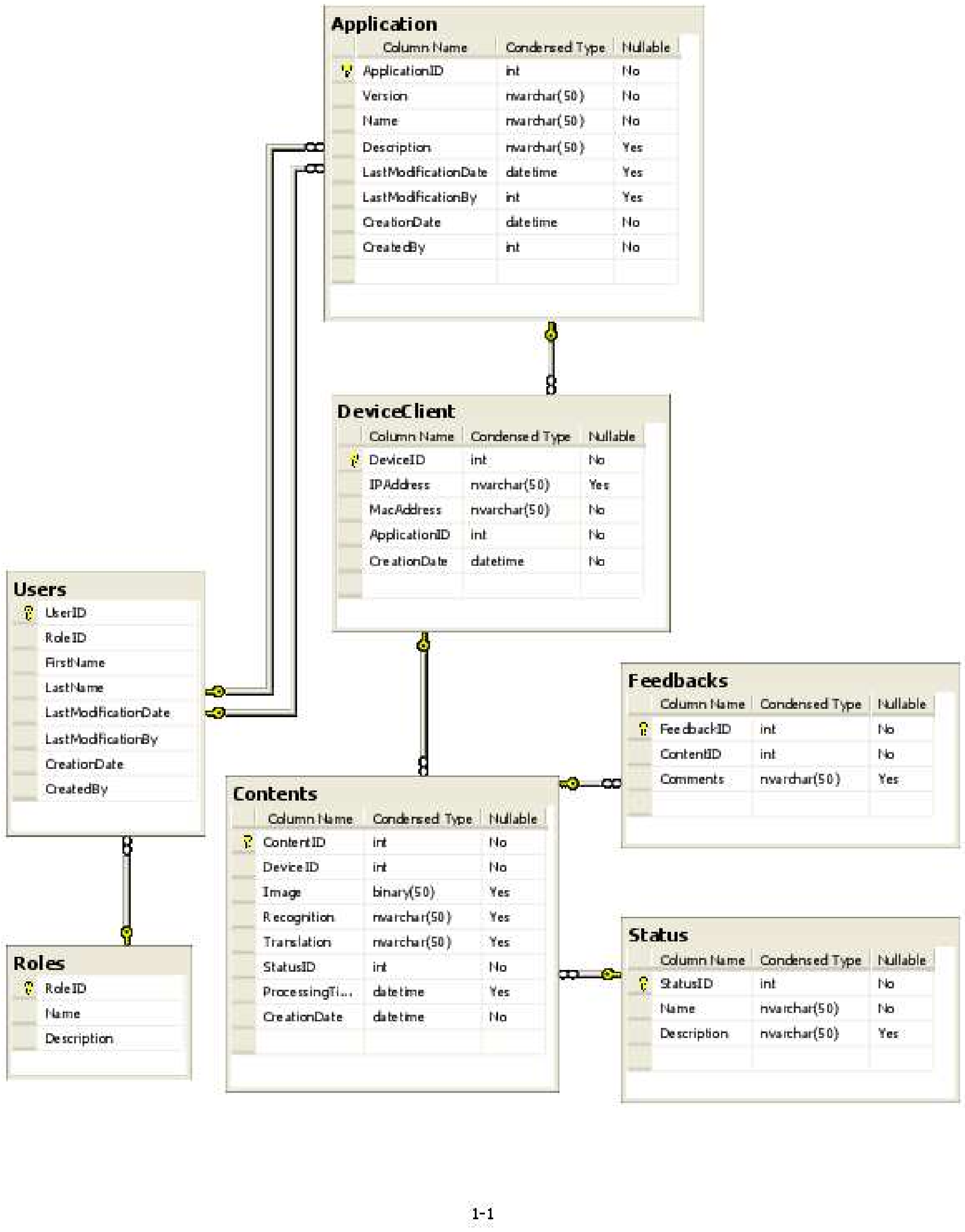
After translation the data in the server, and before it is sent back to the client-side application, it passes by the final module, the Content Storing module, which is responsible of storing some data about the user request, image, recognized and translated text into a SQL database. This information will be taken into account for further investigation and to study the errors made by the system in order to find a way to improve the quality of the system.

For this module to work, it needs the user's permission. It is important to ask the users if they would like to participate in the process of improving the quality of our system by giving us the permission to store some data from their request. This permission can be gained by enabling the "Help in improving our system" check box in the client-side application.

First of all, this module stores data about the user request, such as the IP address, Application version, and the date of the request. This information is stored in the DeviceClient table which is represented in the UML design in Figure 5.2. Then, for each client/device, the data (the image) is stored in the server. Besides from that, this module also stores the recognized text which is the output of the recognition module, as well as the translated text which is the output of the translation module. Additionally, the complete processing time is also stored. This information is shown in the Content table 5.2. Finally, the client can send a feedback about the usage of the system or any suggestions or improvements. This information is stored in the Feedback table.

In the Application table we store the version of the mobile application and some description about it. This tables is filled by us every time a new version of the application is implemented. The user and roles tables are used to store credentials of the user who add ore modify the application version. These tables are completely controlled by us. However, when a client make a request, the ID of the version of the application that he is using is stored in the DeviceClient table.

The Status table is used to indicate the status of the process. Depending on each request, a content record might have failed, succeeded, or interrupted by the user or because of a bad Internet connection. It is important to know the reason when investigating the errors.



Application

Column Name	Condensed Type	Nullable
ApplicationID	int	No
Version	nvarchar(50)	No
Name	nvarchar(50)	No
Description	nvarchar(50)	Yes
LastModificationDate	datetime	Yes
LastModificationBy	int	Yes
CreationDate	datetime	No
CreatedBy	int	No

DeviceClient

Column Name	Condensed Type	Nullable
DeviceID	int	No
IPAddress	nvarchar(50)	Yes
MacAddress	nvarchar(50)	No
ApplicationID	int	No
CreationDate	datetime	No

Users

UserID
RoleID
FirstName
LastName
LastModificationDate
LastModificationBy
CreationDate
CreatedBy

Feedbacks

Column Name	Condensed Type	Nullable
FeedbackID	int	No
ContentID	int	No
Comments	nvarchar(50)	Yes

Contents

Column Name	Condensed Type	Nullable
ContentID	int	No
DeviceID	int	No
Image	binary(50)	Yes
Recognition	nvarchar(50)	Yes
Translation	nvarchar(50)	Yes
StatusID	int	No
ProcessingTi...	datetime	Yes
CreationDate	datetime	No

Roles

RoleID
Name
Description

Status

Column Name	Condensed Type	Nullable
StatusID	int	No
Name	nvarchar(50)	No
Description	nvarchar(50)	Yes

Figure 5.2: Database design for the Image Translation mobile application

Chapter 6

Concluding Remarks

In this work we have discussed an Image Translation system, that takes an Arabic text image as an input and returns the translation of the recognized text in English. This kind of systems, is typically based on a concatenation of two systems: a Handwritten Text Recognition (HTR) system and a Machine Translation (MT) system. In the case of handwritten recognition of text images, our work has focused on the use of the *embedded Bernoulli (mixture) HMMs (BHMMs),* that is, embedded HMMs in which the emission probabilities are modeled with Bernoulli mixtures. These models have shown improvements among other systems in different corpus for Arabic Text. In the case of Arabic text translation, we discuss three different state-of-the-art translation models: the Standard phrase-based model, the Hierarchical phrase-based model, and the Bilingual N-gram models. However our work has only focused on the phrase-based log-linear translation models, In particular, we used Moses toolkit to carry out our experiments.

Experiments in this work were carried out on the Arabic-English parallel corpus provided by the Linguistic Data Consortium (LDC) on the NIST Open Handwriting Recognition and Translation evaluation (OpenHaRT). The data is collected from both 2010 and 2013 evaluations. Both evaluations focus on core recognition and translation technologies for document images containing primary Arabic handwritten scripts.

All results obtained in this work have been submitted to the NIST OpenHaRT 2013 evaluation. Our submissions included systems for both transcription and translation. Specifically, two systems were submitted for the DIR task, one system for the DTT task, which followed both constrained and unconstrained training conditions, and three systems for the DIT task. Very competitive results were achieved.

Finally, the idea of an *Arabic Transcription and Translation system* for mobile applications is presented and discussed. We explored a client-side application for Android mobiles, and also a server-side application. The client-side application is responsible of sending an Arabic text image taken by a user to the server-side application by sockets. The image then processed, recognized, translated, and finally, the translated text is return back to the client.

Bibliography

[1] L.M. Lorigo and V. Govindaraju. Offline Arabic Handwriting Recognition: A Survey. 28(5):712–724, May 2006.

[2] Stefan Jaeger David Doermann, David Scott Doermann. *Arabic and Chinese handwriting recognition*. Springer-Verlag Berline Heidelberg, 2008.

[3] Mohamed Cheriet. Visual recognition of arabic handwriting: challenges and new directions. In *Proceedings of the 2006 conference on Arabic and Chinese handwriting recognition*, SACH'06, pages 1–21, Berlin, Heidelberg, 2008. Springer-Verlag.

[4] Badr Al-Badr and Sabri A. Mahmoud. Survey and bibliography of arabic optical text recognition. *Signal Processing*, 41(1):49 – 77, 1995.

[5] I. S. I. Abuhaiba, S. A. Mahmoud, and R. J. Green. Recognition of handwritten cursive arabic characters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16:664–672, June 1994.

[6] Stephan Jonas Matrikelnummer. Improved Modeling in Handwriting Recognition, jun 2009.

[7] Yi Chang, Datong Chen, Ying Zhang, and Jie Yang. An image-based automatic arabic translation system. *Pattern Recognition*, 42(9):2127–2134, September 2009.

[8] Adrià Giménez, Ihab Khoury, and Alfons Juan. Windowed Bernoulli Mixture HMMs for Arabic Handwritten Word Recognition. In *Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Kolkata (India), November 2010.

[9] Ihab Khoury, Adrià Giménez-Pastor, Jesús Andrés-Ferrer, and Alfons Juan-Císcar. Arabic Printed Word Recognition Using Windowed Bernoulli HMMs. In *Proc. of the 17th Int. Conf. on Image Analysis and Processing (ICIAP 2013)*, Naples (Italy), September 2013. Accepted.

[10] Fouad Slimane, Slim Kanoun, Haikal El Abed, Adel M. Alimi, Rolf Ingold, and Jean Hennebert. ICDAR 2011 - arabic recognition competition: Multi-font multi-size digitally represented text. pages 1449–1453. IEEE, September 2011.

[11] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, and R. Zens. Moses: Open source toolkit for statistical machine translation. In *Annual meeting-association for computational linguistics*, volume 45, page 2, 2007.

[12] A. Tong, M. Przybocki, V. Maergner, and H. El Abed. Nist 2013 open handwriting recognition and translation (openhart'13) evaluation. *Proceedings of the NIST 2013 Open Handwriting and Recognition Workshop*, 2013. In press.

[13] Joshua T. Goodman. A bit of progress in language modeling. Technical report, 2001.

[14] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, 1993.

[15] S. Young et al. *The HTK Book*. Cambridge University Engineering Department, 1995.

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

[17] Jesús Andrés Ferrer. *Statistical approaches for natural language modelling and monotone statistical machine translation*. PhD thesis, UNIVERSIDAD POLITÉCNICA DE VALENCIA, Valencia, Spain, February 2010.

[18] F. J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004.

[19] D. Vilar, D. Stein, M. Huck, and H. Ney. Jane: Open source hierarchical translation, extended with reordering and lexicon models. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, page 262–270, 2010.

[20] J. B. Marino, R. E. Banchs, J. M. Crego, A. de Gispert, P. Lambert, J. A. R. Fonollosa, and M. R. Costa-jussà. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549, 2006.

[21] Josep Maria Crego and José B. Mariño. Improving statistical MT by coupling reordering and decoding. *Machine Translation*, 20(3):199–215, July 2007.

[22] Josep M. Crego, François Yvon, and José B. Mariño. Ncode: an open source bilingual n-gram SMT toolkit. *The Prague Bulletin of Mathematical Linguistics*, 96(-1):49–58, October 2011.

[23] NIST. NIST 2010 Open Handwriting Recognition and Translation. Technical report, The National Institute of Standards and Technology (NIST), 2010.

[24] A. Giménez and A. Juan. Embedded Bernoulli Mixture HMMs for Handwritten Word Recognition. In *Proc. of the 10th Int. Conf. on Document Analysis and Recognition (ICDAR 2009)*, pages 896–900, Barcelona (Spain), July 2009.

[25] Ihab Khoury, Adrià Giménez, and Alfons Juan. Arabic Handwritten Word Recognition Using Bernoulli Mixture HMMs. In *PICCIT '10*, Hebron (Palestine), March 2010.

[26] Ihab Khoury, Adrià Giménez, and Alfons Juan. Arabic handwriting recognition using bernoulli hmms. In Volker Märgner and Haikal El Abed, editors, *Guide to OCR for Arabic Scripts*, pages 255–272. Springer London, 2012.

[27] Adrià Giménez, Ihab Khoury, Jesús Andrés-Ferrer, and Alfons Juan. Handwriting word recognition using windowed bernoulli hmms. *Pattern Recognition Letters*, 2013. In press.

[28] Nobuyuki Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans. on Systems, Man and Cybernetics*, 9:62–66, 1979.

[29] A. Stolcke. SRILM-an extensible language modeling toolkit. In *Proc. of the Inter. Conf. on spoken language processing*, volume 2, page 901–904, 2002.

[30] Owen Rambow Nizar Habash and Ryan Roth. Mada+tokan: A toolkit for arabic tokenization, diacritization, morphological disambiguation, pos tagging, stemming and lemmatization. In Khalid Choukri and Bente Maegaard, editors, *Proc. of the 2nd Int. Conf. on Arabic Language Resources and Tools*, Cairo, Egypt, April 2009. The MEDAR Consortium.

[31] Andreas Eisele and Yu Chen. Multiun: A multilingual corpus from united nation documents. In Daniel Tapias, Mike Rosner, Stelios Piperidis, Jan Odjik, Joseph Mariani, Bente Maegaard, Khalid Choukri, and Nicoletta Calzolari (Conference Chair), editors, *Proc. of the Seventh conf. on Int. Language Resources and Evaluation*, pages 2868–2872. European Language Resources Association (ELRA), 5 2010.

[32] Ted corpus in the iwslt 2011 evaluation campaign, http://iwslt2011.org/doku.php?id=06_evaluation, 2011.

[33] X. Ma. Champollion: A robust parallel text sentence aligner. In *LREC 2006: Fifth International Conference on Language Resources and Evaluation*, page 489–492, 2006.

[34] Guillem Gascó, Martha-Alicia Rocha, Germán Sanchis-Trilles, Jesús Andrés-Ferrer, and Francisco Casacuberta. Does more data always yield better translations? In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 152–161, Avignon, France, April 2012. Association for Computational Linguistics.

[35] The translectures-upv team. the translectures-upv toolkit (tlk). http://translectures.eu/tlk. http://www.translectures.eu/tlk/citing-tlk/, 2013.

[36] Qin Gao and Stephan Vogel. Parallel implementations of word alignment tool. In *In Proc. of the ACL 2008 Software Engineering, Testing, and Quality Assurance Workshop*, 2008.