Hana Demas

Making Amharic to English Language Translator for iOS

Helsinki Metropolia University of Applied Sciences
Degree Programme In Information Technology

Thesis

Date 5.5.2016

Author(s)	Hana Belete Demas
Title	Amharic To English Language Translator For iOS
Number of Pages	54 pages + 1 appendice
Date	5 May 2016
Degree	Information Technology Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Petri Vesikivi

The purpose of this project was to build a language translator for Amharic-English language pair, which in the beginning of the project was not supported by any of the known translation systems. The goal of this project was to make a language translator application for Amharic English language pair using swift language for iOS platform.

The project has two components. The first one is the language translator application described above and the second component is an integrated Amharic custom keyboard which makes the user able to type Amharic letters which are not supported by iOS 9 system keyboard. The Amharic language has more than 250 letters and numbers and they are represented using extended keys. The project was implemented using the Swift language.

At the end of the project an iOS application to translate English to Amharic and vice versa was made. The translator applications uses the translation system which was built on the Microsoft Translator Hub and accessed using Microsoft Translator API. The application can be used to translate texts from Amharic to English or vice versa.

Keywords	API, iOS, Custom Keyboard, Swift, Microsoft Translator Hub

Contents

1. Introduction

With the advent of smartphones and with the rapid development of mobile application development, users can download useful applications which help them facilitate their day to day life. Translator applications are among those applications that make communication and information search in different languages easier. With the help of translator applications people can at least get a general understanding of a foreign text or speech. This is particularly useful for developing countries such as Ethiopia with scare written resources in science and technology.

The Ethiopian language Amharic is a Semitic language which is the national language of The Federal democratic republic of Ethiopia. It is spoken by over 17 million people in Ethiopia, which is 32.7% of the countries population. The Amharic language has a semi-syllabic writing system. [2, 1.] The Amharic Alphabet has 33 consonant symbols with 7 variations. The variations are formed by coupling these consonants with the Vowels. In total there are over 250 letters and numbers in the Amharic writing system. [17].

Machine translation (MT) is the process of translating a source language to a target language using computers the most popular, low cost and general purpose translators in the market, such as Google and Microsoft translators are based on a type of MT called statistical machine translation which produces language rules by evaluating the patterns of text from a huge number of documents from both source and destination languages using a statistical approach [1, 2.]

The Microsoft translator API is a service provided by Microsoft to translate texts in a foreign language and carry out other language processes such as speech recognition and text detection. The Microsoft translator is based on a statistical approach and provides a free translation service for 52 languages as of March 2016 [10]. Microsoft also has a service called Microsoft Translator Hub which is used to build a custom translation system for users for their own needs. This include building a translation system for a language not supported by the API of Microsoft. [19].

Since the structure of the Amharic language is not well studied in comparison with the structure of other international languages such as English, it is hard to build a rule-based translation system. Most of the researches done in the field on the language are

also based on a statistical approach. [6, 2] One of the main reasons the Microsoft translator Hub was built was to provide a simple platform for building a translation system for languages such as Amharic. The Hub builds a translation system with zero knowledge of the structure of the Amharic language.

The purpose of this project was to build a translator application which can be used by people who want to translate texts from Amharic into English and vice versa. The target group was anyone with an understanding of the Amharic alphabet and who wants to learn English or Amharic. As the Amharic alphabet is different from English, one should know how to read an Amharic text to grasp the meaning.

The goal of this project was to build a translator iPhone application for the Amharic-English language pair. At the start of this project the Amharic language was not supported by any translation software. The Amharic characters are not supported by the iPhone operating system and therefore a custom Amharic keyboard to represent the Amharic character was built and it is the Integral part of the application. Anyone who has downloaded the application can also use the keyboard in other applications on the Phone. The translator application is based on a translation system built on the Microsoft translator platform called Microsoft Translator Hub. The Microsoft translator API then was accessed using swift language to fetch the translation. The application was build using Xcode and Sketch 3 to design the small icons used in the application.

Currently the application translates Amharic to English and English to Amharic. It can easily be extended to other languages as well by adding more projects to the Hub for the required language pairs. Text translation is the only text processing functionality supported since the Hub does not support functionalities such as text detection and uttering of text, which the normal Translator API supports.

This thesis is structured as follows. The second chapter will discuss the theory of translation and the next one will discuss the Microsoft Translator Hub and how the translation system was built. In the fourth section the implementation of the application

will be discussed and later the iOS custom Amharic keyboard. Finally there is a section to compare the translation software on the market and discussion and conclusion sections.

2. Machine Translation (MT)

Before the advent of digital computers tasks which require Intelligence such as the translation of a language to another language were thought to be impossible to be implemented by a machine. [1]. Machine translation is the use of computerized machines to translate a source language into a target language. In 1970s the foundation of the first MT was laid in. Machine translation nowadays is mainly based on a statistical system which can produce a language rule based on large corpus of translated text, instead of studying the language rules of every language and develop an algorithm to implement these rules [1, 4-5]. The first is called Statistical Machine Translation (SMT) and the latter is called Rule-based Translation System (RBMT). A hybrid translation system is a system which combines SMT and RBMT. [3, 1.]

The quality of a machine translation is measured based on the corresponding human translation [3, 1]. This means that translating a text is not merely substituting the translation of individual words which make the whole text. It should also take into consideration the context of the sentence. To accomplish this it requires deep knowledge of the grammar, sentence structure and semantics of both the target and source language. Additionally, as not all people will come up with exactly the same translation of a given sentence it is a difficult task for a machine to produce a single translation which meets the requirements of all users. Due to this translation systems are more efficient when they are used for a specific domain such as Medicine, Culture or Engineering than a general purpose translator [3, 3]. Recent trends are in building domain-specific MT systems with speech recognition, especially for hand-held devices. [1, 4.]

2.1 Statistical Machine Translation (SMT)

SMT is a translation system which produces language rules based on a large amount of translated text using statistics and probability rules. It is based on the assumption that it is hard to fully analyze the rules of a given language and distill them into a set of rules as language is constantly changing and there are grammatical exceptions in every language which cannot be incorporated into the language rules. SMT is the most researched translation mechanism and also widely used in the marketplace .The first pure SMT Company is Language Weaver and SMT is also the method used by the free online translation systems of Google and Microsoft. [1, 18]. Even though most of the MT systems do not give a good translation of a sentence into most languages, they give the basic understanding of the whole text for the foreign language speaker [7, 3.]

In most SMT systems a text is broken into sentences which are strings of words with punctuation. Individual words are separated using a method called tokenization. The distribution of a word in a text is different since some words occur more frequently than others. Words can be categorized by their parts of speech or meaning. Some languages exhibit rich inflectional morphology, which results in a large vocabulary and in languages such as Chinese it is hard to identify what a word is as the writing system does not use space to separate them. A text can be also be analyzed in a hierarchical way using clauses, phrases and dependencies between distant words that may be best represented graphically in tree structures. SMT systems are based on parallel corpora which are texts, paired with a translation into another language. Parallel corpora are produced by extracting web documents such as HTML and carrying out document and sentence alignment on the extracted text. [1, 33-34].

The extracted text then has to go through text conversion, trimming before and after alignment at document level, sentence splitting before alignment at sentence level, sentence aligning and tokenization. Depending on the SMT machine the processed text has to be changed to the write document format such as PDF before feeding it into

the SMT. The SMT system trains this processed bilingual corpora and associates a probability to a given translation by counting the co-occurrence of it in the whole data. Increasing the size of the corpora increases the accuracy of the translation probability. [7, 8.] All the texts for a given language also should be in the same font as the SMT is not smart enough to identify the same word in two different fonts. The SMT can be modelled using the following diagram. The components are Translation model, Language model and Decoder.

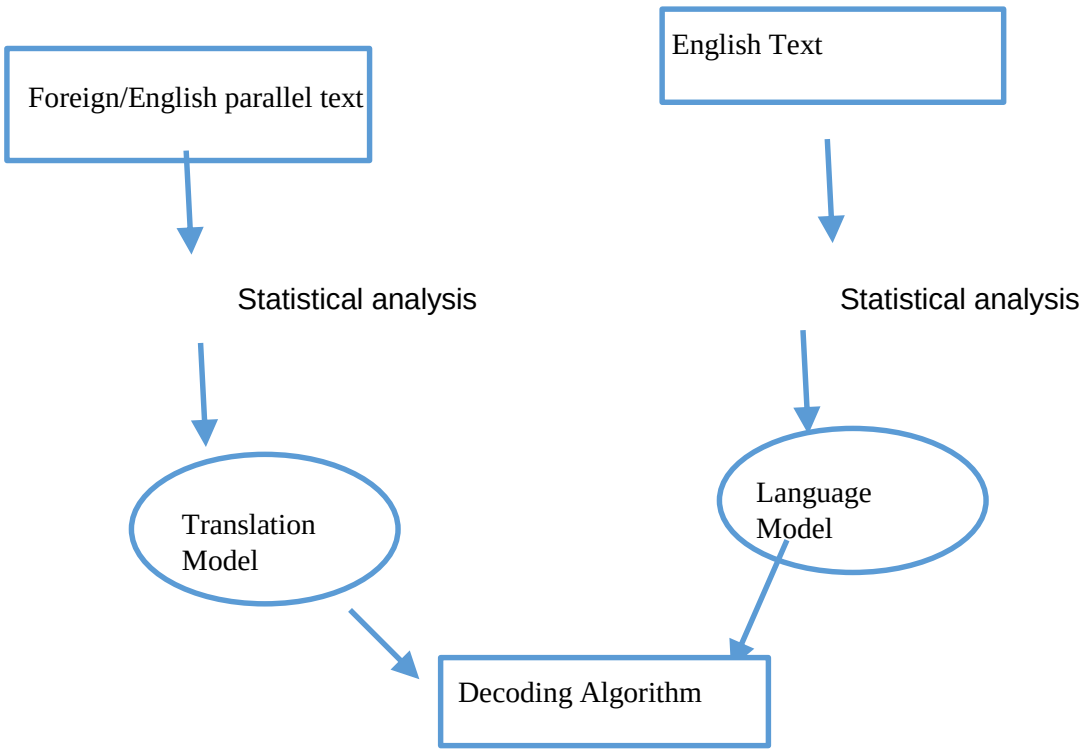


Figure 1 Architecture of SMT (extracted from [5, 5])

SMT system can be modeled using the Figure 1. First a parallel document will be statistically analyzed and aligned sentence, phrase or word based depending on the model and a translation model will be produced from this statistical analysis. Also the source language will go through a statistical analysis to have the model of it. Finally the

decoding algorithm produces a translation of the source language into the target language using the Translation and Language Models.

2.1.1 Parallel Corpora

A parallel corpora is a collection of texts of one language with its corresponding translation into the target language. In most cases they are acquired from websites which are translated into multiple languages. However in most cases the web documents are not direct translations from the other language. They are modified to suit the varied backgrounds of the audiences. This creates a problem called a document alignment issue. After aligning the Parallel Corpora documentwise it has to go through a process called sentence alignment [1, 55-56].

Sentence alignment can be formally defined as follows:

Given the sentences f1 ,..., fnf in the foreign language to the sentences e1 ,...,ene in English, a sentence alignment S consists of a list of sentence pairs s1, ..., sn. Each sentence pair si is a pair of sets si = ({ fstart-f(i), ..., fend-f(i)}, {estart-e(i), ..., eend-e(i)})

The alignment type is defined as

type = |{ fstart-f(i), ..., fend-f(i)}| — |{estart-e(i), ..., eend-e(i)|

The type is 1-1 if one foreign sentence is aligned to one English sentence, 1-2 if one foreign sentence is aligned to 2 English sentences and 1-0 if there is no aligned English sentence [1, 56].

2.1.2 Models of Translation

Word-based Model

A word in one language can have multiple translations in another language. For example the German word “Haus” can have the English translation of “house’,

“building”, “home”, “household”, “shell”. The SMT counts the occurrence of each English word as the translation to the word “Haus” i.e. the method takes into consideration only the possible translations and their corresponding occurrences.The problem can be modelled using the following probability distribution function

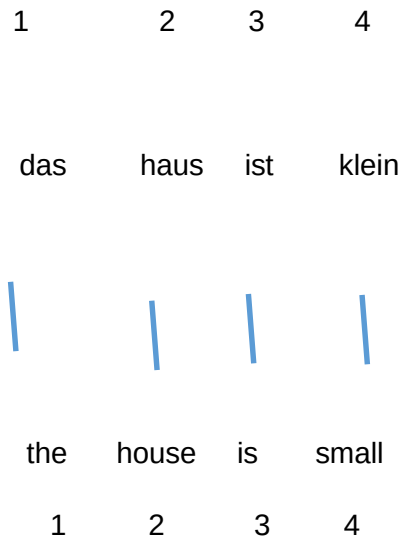
pf :e→pf(e) which is called lexical translation probability distribution.

Let us say for the above example the translations have the following occurrences in a given text

Translation of Haus	Count
house	8000
building	1600
home	200
household	150
shell	50

From the definition of the probability distribution function, the probability of a given event should be a number between 0 and 1 and the summation of all occurrences should be 1.Therefore from the above data the probability of each occurrence can be calculated by dividing the occurrence of the English word with the total occurrence of the word “Haus”. For example pHaus (house) = 0.8 . The SMT then uses the conditional probability t (e|f), which is probability of translating a foreign word f into an English word e. These probabilities are represented by T- Tables (Translation Tables). For example the translation of the German sentence “das Haus ist klein” Each foreign language word is aligned to its translation in English. [1, 81–82.]

The word positions are numbered.

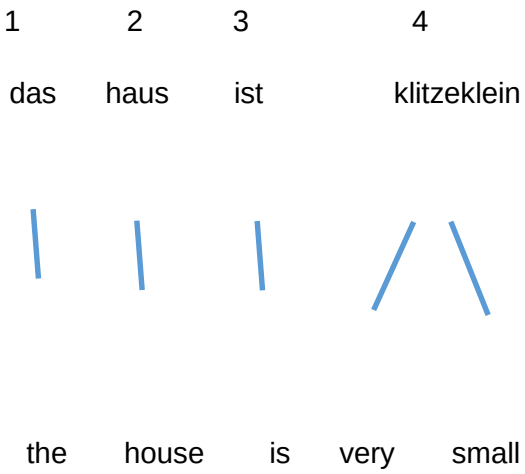


To accomplish this there is a concept called the alignment function which is the making of the foreign word at position j with the English word at position I. And defined as a: i → j.

For the above example the alignment function would be

a : {1 → 1, 2 → 2, 3 → 3, 4 → 4}

Exceptions to this are in most languages that the word ordering is different. For example the position of the verb in German and English is different in different sentences. In this case the words in a sentence need to be reordered. In other cases one word in source language might map to two words in the target language or vice versa. An example of this is the German word “klitzeklein” has the English translation of very small. In this case the making function will look like the following: [1, 84-85.]



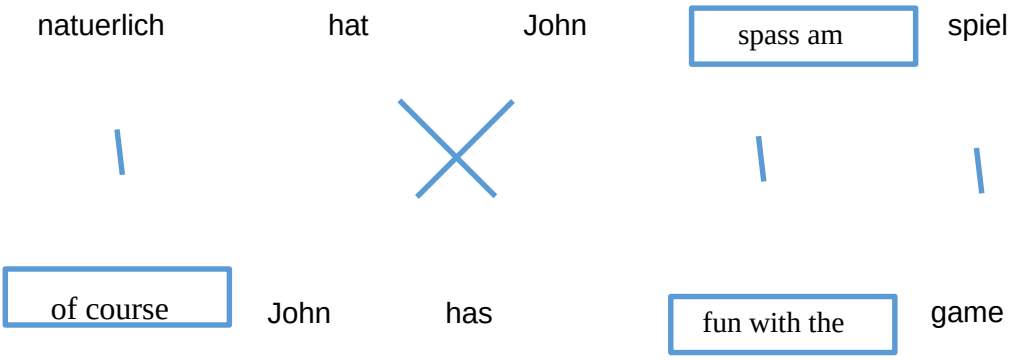
a : {1 → 1, 2 → 2, 3 → 3, 4 → 4, 4 → 5}

Other cases might be words which will be dropped and have no mapping in the target language and a word may be inserted in the target sentence with no mapping to the source and still give the same meaning as the whole aligned translation. The most known word-based model is the IBM 1 model and higher IBM models such as IBM 2,3,4,5. IBM1 model produces a number of different translations for a sentence, each with a different probability. The word alignment in this model is based on an algorithm called expectation maximization algorithm (EM algorithm). The EM algorithm is composed of four steps .The first step is initialization of the model with uniform distribution, which means assuming that each possible translation of the word has equal probability. The second step called the Expectation step will find the most likely translation of the word. In the third step called the Maximization step a weight to every possible alignment is given. The last step is to iterate through the second and third steps until convergence. Perplexity is the measure of the convergence of these iterations. The Perplexity decreases or stays the same after successive iterations. [1, 86-90]

Phrase-based Model

Phrase base model translates a small sequence of words at a time and it is used by the best performing statistical machine translation systems on the market. As discussed in the section above word based translation fails in situations where one word in source language doesn't make to exactly one word in the target language. In the phrased based model the sentence is broken down in segments called phrases and reordered according to the grammatical role of the language. [1, 128]

Example:



The word “Phrase” in the phrased-based model is different than its linguistic definition. In the Phrased-based model, words in a phrase are grouped based on context, i.e. how often the word will be followed by the other words in the group. This model has the advantage of learning the translation of a longer and longer phrase from a big corpora of text and even a possibility of learning the whole sentence which cannot be accomplished using the word based model.

Phrase-based models have been used as part of a syntax-based translation model where each phrase corresponds to syntactic subtrees in a parsed sentence. The Syntax-based model as proposed by Kenji Yamada and Kevin Knight (2010) was designed to incorporate the structural difference of sentences in different languages into the Word-based Model. In this model a sentence preprocessed by a syntactic parser is used as an input. [14, 1.]

Phrased-based models use the automatically-generated word-level alignments from the IBM models to extract phrase-pair alignments. They are considered an intermediate step between word-based and more linguistically-oriented translation models. The fundamental units of the translation model are the phrase-pair alignment probabilities and its performance outperform the syntax-based translation model.

The Phrase based model can be mathematically described as follows:

Suppose E is an English sentence, and F is a foreign sentence. If we segment the source sentence F into a sequence of I phrases

$$F = f1, f2,... fI,$$

Then each phrase fi in F is translated into an English phrase, called ei, using a probability distribution φ(fi|ei). The probability of all segmentations is considered to be equal.

$$P (F | E) = P (f 1I | e I1)$$

$$P (e I1) \cdot P (e I1 | f 1I) P (f 1I)$$

Therefore the most likely translation sentence Eˆ is

$$E = = \text{argmax} \prod P(ei) \cdot \varphi(fi|ei) E \; i=1$$

This is called the noisy channel approach. The other and often used approach is called a log-linear framework [1, 132].

Syntax-based model

The Syntax-based model is used to incorporate the structural difference of sentences in different languages into the Word-based model. The model accepts a parse tree as an input and in the processing channel operations such as reordering child nodes, inserting extra words at each node, and translating leaf words are performed on each node of the parse tree. The translation between SVO languages (Subject-Verb-Object) such as English and SOV languages (Subject-Object-Verb) such as Japanese is modelled using a reorder operation. The insertion operation is used to incorporate syntactic cases. One example of this is the use of the structural position to specify the case in English sentences and case-marker particles in the Japanese language.

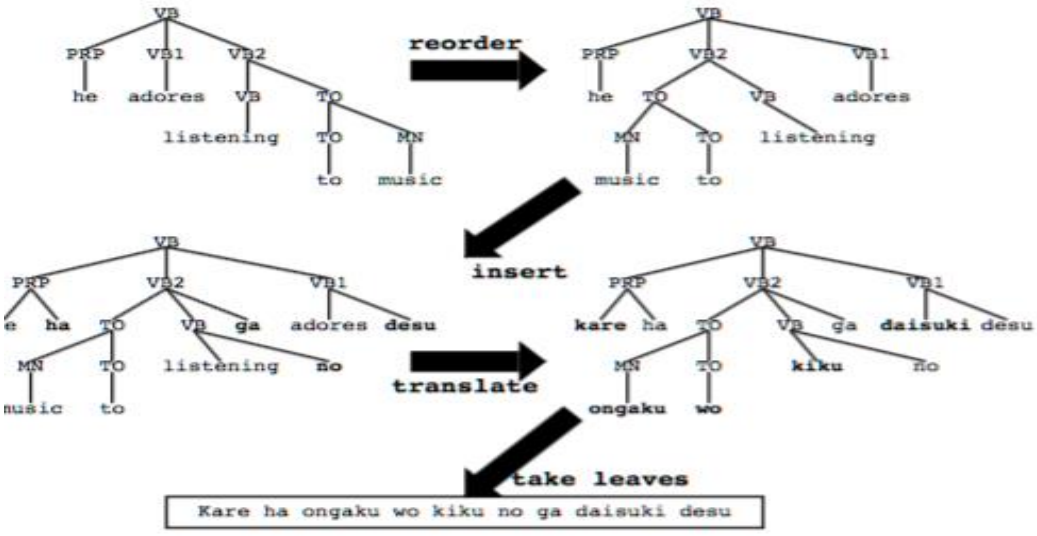


Figure 2. Insertion, reorder and translate operations (copied from [16, 11]).

The model proposed by Kenji Yamada and Kevin Knight (2010) is depicted in the diagram above, assuming an English parse tree is fed into a noisy channel and that it is translated into a Japanese sentence. The first step is to reorder child nodes on each internal node. A node with N children has N! possible reordering. The reordering is described in a table called r-table and the reordering is influenced only by the number

of child-nodes. In Figure 2 the Verb base form (VB) has children node sequence node has a child sequence PRP (Personal pronoun)-VB1-VB2 with an ordering probability of 0.723. In the second level VB-TO (to) into TO-VB, and TO-NN (Noun) into NN-TO are reordered with a combined probability of 0.484. The insertion and translation are also done as depicted in Figure 2. The insertion probabilities are described using n-table which describes the probability of insertion on a phrase at a parent, left or right node. The translation process is dependent on the word itself and not on the context. The t-table describes the translation probability of a word. The final output of the system is a string which is different from the parse tree input.

After aligning the sentence word or phrasewise, it will go through a process of decoding, which is a process of finding the best scoring translation among all possible translations. The decoding is based on a technique called heuristic search, which is a method which does not guarantee it finds the translation with a high probability all the times, but tries to find it often enough. The fluency of this translation is increased by using a language model. A language model is a measure of how probable a given sequence of a word will be uttered in the language. N-Gram language model is the best-known one.

2.1.3 Language Models

A language model helps to determine how likely is a given sentence to be uttered by the speakers of the language. For example how probable is an English speaker to say “the house is small” instead of “small the is house”. This helps determine word order [1]

plm (the house is small) > plm(small the is house)

Also in terms of work choice, For Example an English speaker is most likely to say ”I am going home” than “I am going house”

plm(I am going home) > plm(I am going house)

The N-Gram language model is one of the simple and robust language models and was first introduced by Jelinek and Mercer. In this model language is broken down into arbitrary sequences of symbols with little consideration to meaning and structure. It is assumed that the nth word w depends only on the history h, of the preceding n-1 words. In other words, for example in the 3-Gram model a word is dependent only on the preceding 3 words and for example the 4th word does not have an effect. A higher N value will produce a more accurate result. However, due to data sparseness a given word may appear less frequently on a large corpora. Due to this tri-Gram models are most often used. To compensate for the sparseness of n-gram counts, various smoothing techniques can be used. [14, 13-15]

2.2 Rule-Based Translation (RBMT)

The classical MT research in the 1970s and 1980s such as SYSTRAN and Eurotra were based on RBMT. RBMT is based on a countless number of dictionaries and built-in linguistic rules. This requires extensive lexicons with morphological, syntactic, and semantic information, and large sets of rules. The RBMT system parses the source text and applies all these language transformation rules and produces a text in the target language. This involves arranging the target text based on its grammatical rules. Improving the quality of an RBMT system is mainly achieved by adding additional sets of dictionaries, which in many cases is a time consuming, expensive and a process that can only be done with a well-trained expert in the field. [7, 5]

Generally building and even improving the quality of RBMT is costly and as every language is constantly changing from time to time updating the rules and dictionaries is a mandatory task. The advantage of RBMT is that one can achieve a higher translation quality than SMT and there is no need to feed a tremendous amount of data as in the equivalent SMT system. However this can only be achieved with a huge amount of money and time with involvement of a lot of linguistic expertise [3, 3-4]. For this reason

the implementation of RBMT is restricted to organizations like the National Security Agency with requires high accuracy of translation.

Direct, transfer, and interlingual are the three historically known RBMT types. The direct approach follows a unidirectional approach where it translates from a source to target language. In the interlingual approach a general representation of meaning that is independent of the language pairs is created. This approach involves two steps for each direction of the language pair, i.e. from the source to the model which represents the meaning, and to the target. In the transfer approach there are three steps aimed at generating some level of syntactic representation for both source and target languages [7, 5].

2.3 Hybrid MT Systems

As mentioned in section 2.1 SMT systems have the disadvantage of the translation not being of good quality due to the difficulty of producing grammatically correct translations, while the RBMT system can produce good translations but requires a tremendous amount of money and skilled technicians and it is difficult to achieve translation convergence. To optimize the high cost of RBMT and the relative low quality of SMT, nowadays hybrid systems are emerging. AppTek, which was launched in 2009, is an example of such systems. Even though the SMT and RBMT systems can be used in different ways to produce the hybrid systems they can be broadly categorized into two approaches. In the first one several existing MT engines are combined in a multi- engine setup, and a decoder for SMT is used to select and combine the best expressions proposed by different engines. One example for such approach is the EuroMatrix project [7, 16].

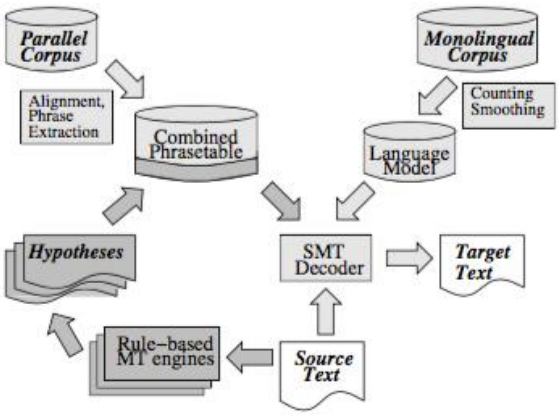


Figure 3. Architecture for multi-engine MT driven by an SMT decoder (copied [18, 2])

In the second approach SMT phrases are fed into a rule-based MT system. This is a variant of SMT where lexical information from rule-based engines is used to increase lexical coverage. The approach investigates how automatically extracted lexical knowledge can be used to increase the coverage of a rule-based MT system as lexical entries can be automatically induced from existing translations of SMT to increase convergence of the rule based system. The other approach of hybrid systems is a linguistically informed rule-based engine which performs translation and the result will be postedited using statistical methods or a linguistic rule is used to guide an SMT system, I.e. a rule-based approach is used to postedit the result of the SMT. [7, 16]

In systems such as AppTek, the rule-based and statistical approach are integrated from the start of the project. Hybrid systems are on ongoing research and they are not prevalent on the market .But the results of researches on the approach have shown that hybrid MT output achieves greater fluency and more aptly translates both "literal and non-literal meanings" of a text. [7, 3].

3. Microsoft Translator Hub

Microsoft translator is an MT-based on SMT which transforms text between languages from existing human translations and uses statistics and machine-learning techniques to produce a translation in the target language. The Microsoft translator is mostly a non-domain-specific general translator which works with any hardware platform and with any operating system. In addition to translating a plain text, the Microsoft translator gives language-related operations such as language detection, text to speech, or dictionary and document translation. [19]

Out of more than 7000 languages spoken all over the world only around 100 are supported by automatic language translation systems on the market. This is a barrier for people from undeveloped countries for sharing global knowledge. Microsoft translator has a platform to build a custom translation for a specific need or to support unsupported languages via cloud and web services. This is helping the world in exchanging experience of speakers of different languages [4]

Microsoft translator HUB is an extension of Microsoft translator platform and service which lets users build their own translation system using their private websites or by feeding a corpora of documents using Microsoft Translator's big data backend. After registering to Windows Azure one can set up a translation project and feed aligned parallel corpora and monolingual dictionaries and the system trains the data and builds a translation system after a request for deployment. The whole process takes a maximum of four working days for not huge amount of data. Once deployed the system is accessible through standard Microsoft translator APIs, using HTTP, AJAX, and SOAP interfaces, as well as a web page widget. The Hub is free of charge and is powered by Windows Azure [19].

Building a translation system on Microsoft translator Hub

At the beginning of this project the Amharic language was among the unsupported languages by the known translation systems. The Microsoft translator Hub is an SMT system developed by Microsoft Research and released for public and commercial use in July 2012 .The Hub allows any individual to build his/her own translation system whether the language pair is supported by the Microsoft translator already or it is among the unsupported ones. The Hub comes with a user guide with a clear and easy steps to build a translation system and an API guide to access the built system later. To use the Hub a workspace should be created with Microsoft Live ID in the Hub platform.

Figure 4 depicts the steps to build a translation system on the Microsoft translator Hub.

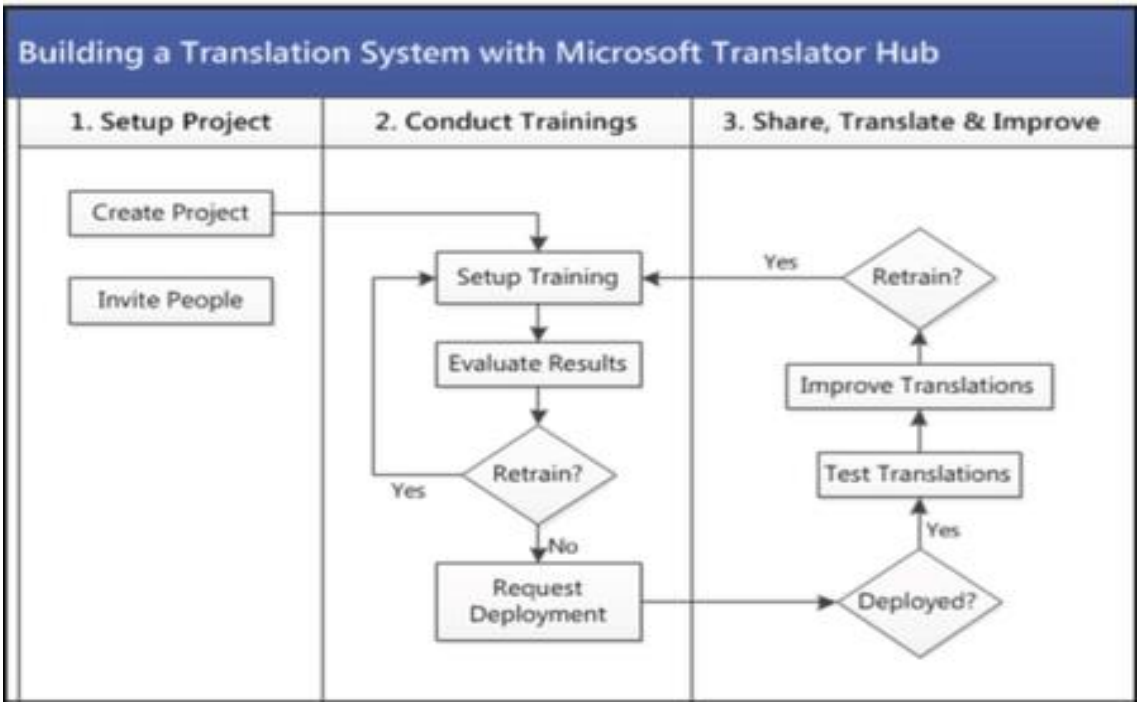


Figure 4. Steps to build translation system on the Microsoft translator Hub (copied from [19])

The first step in building a translation system on the Hub is to create a project on the workspace one has created. Any number of people can be invited to collaborate on the project. The next step is to set up a training. The training is set up by giving parallel and monolingual documents in the format of XLIFF (XLF, XLIFF), TMX , LCL (LocStudio generated files), Microsoft Word documents (DOCX), Adobe Acrobat files (PDF), HTML files (HTML, HTM), UTF-16 and UTF-8 encoded text files (TXT, ALIGN). The files should also follow the file naming convention described in the user guide [19, 8].

After setting up the training, it takes 2-12 hours for the Hub to train the data depending on the how difficult the translation for a given language pair is and the amount of data

fed into the Hub and many other factors. The user will be notified by email that the training is completed. The user then evaluates the result and sets up a new training with additional sets of data or by adjusting the test and tuning data sets if the result is not satisfactory or request deployment if satisfied by the result. The deployment takes 2-3 working days and the hub notifies the user in the same way as for the training. After the deployment there is an interface to test the translation system and evaluation can be done using this interface. If the results are not satisfactory, the user can initiate the training again. [19]

In a given workspace one can train any number of projects but only one system per project can be finally deployed. System can be trained multiple times before deployment until a satisfactory result is found. After deployment the Hub provides with a test interface as mentioned above. Using this interface one can test the actual translation the system makes and it also provides one with a category ID for later use while accessing the system through the API. Additionally there is a link to the API guide in the test interface. All the project deployed in a given workspace will have the same category ID and this makes it easy while accessing the system through API. [20]

Since the Amharic language is not supported by the Microsoft translator API, a workspace on the Hub was set up and six projects were created. The projects are Amharic to English, English to Amharic, English to German, German to English, Amharic to German and German to Amharic. The German language was added to compare the results of the training between the already existing language pairs on the Microsoft translator and also to see how the amount of parallel corpora affects the result of the translation system.

The translation system works based on translation logic which it learnt from previously translated documents that are stored in a statistical model. There is an option of using this existing model (The Microsoft Model) or produce one’s statistical model from the documents one has uploaded in your training. The Hub also allows to choose which

translation domain to train for such as Medicine, Technology. All projects created were general purpose translations systems.

For the project parallel corpora were collected from OPUS which is an open content package where one can get the aligned documents for free. The document found were mostly from the Muslim Bible called Quran in the format of TMX (Translation Memory Exchange) and some small global news and related documents for the Amharic English pair and for the Amharic German pairs. But for the English German pair many documents from different domains were collected. The only problem to the English to German Corpora was some of the TMX files were big in size and the maximum size that can be uploaded to the Hub is 100MB. For this reason a TMX editor was used to split the files into the appropriate size.

The system then learns how words, phrases and sentences are mapped and learns the context depending on the surrounding phrases between the two languages. This means that the definition of a word in a sentence is not always the same, but it is dependent on the surrounding words and phrases. One might get different definitions for the same word depending on the surrounding phrases. The next step is to set up the training data sets which are monolingual and parallel documents that the Hub uses as a basis to build the translation system. The Hub requires another sets of data called a tuning data sets which are used to tune the translation system for optimal results. For both the test and tuning data sets the default documents were used. The test data sets are used by the Hub to evaluate the BLEU (Bilingual Evaluation Understudy) score of the translation system.

The BLUE score is a measure of the translation quality of the system. The reference to this score is the test data set. The score ranges between 0 and 100. A BLUE score of 0 means there is no data in the training which matches with the set of test data. The test set which is only composed of bilingual data has no influence over the translation quality. The amount of test data needed is 2500 aligned sentences [19, 38].

The Hub shows the aligned text count on the top while training the data. The minimum number of aligned sentences for the system to pass the training is 10,000. The Hub then reads one sentence with its corresponding translation and aligning words and phrases in these two sentences to each other sequentially.

As mentioned above the tuning data sets are among the data set the Hub uses. The tuning data sets are used to adjust all parameters and weights of the translation system to the optimal values. This data sets have a big influence over the translation quality. The system then tries to find translations which are closest to the samples provided in the tuning dataset. Manually choosing the tuning data set will give a better quality translation than choosing the default one which the Hub selects randomly. Choosing a sentence whose length is between 8- 18 sentences and fluent sentences are recommended by the Hub for optimal results. A fluent sentence is a sentence which does not contain table cells, poems and lists and written as a grammatically correct full sentence.

The test data sets are used by the Hub to compute the BLUE score. The test data sets are chosen where the target language sentences are the most desirable translations of the corresponding source language sentences in the pair. It is also possible to upload monolingual phrases as the dataset called dictionary. It is used to define proper names and product names and will appear in the translation exactly as they are defined. Uploading a dictionary is optional.

One interesting feature of the Hub and Microsoft translator API is the Collaborative Translation Framework. It is a built-in server-side storage of sentence pairs in both source and target languages. The sentences stored in this server can be used by the API as immediate sources for subsequent calls to the API. Another useful use of this sentences is that they can be used by the Hub as a new training data set to improve the translation system. The community who makes use of the translation system through websites, applications and infrastructures can contribute for the betterment of the system through this framework. [19, 14.]

For the sentences produced by the Collaborative Translation Framework to override the original translation by the system, the editing should be done with a person with authority of 5 or higher (in the API: rating parameter is >= 5 (which is the parameter for the addTranslation() method of the API)). The Microsoft translator API has the addTranslation () method to make contribution to the sentence storage server. Later on this can be used by the hub as a training data set.

For the Amharic English translation system the system extracted 114,092 and the total aligned sentence count was 103,443. The BLUE score for the system was 7%. Using the same sets of data the total extracted sentence count for the English Amharic system was 112,896 and the aligned sentence count was 102,262 and the BLUE score was 16.6%.

In the “Evaluation” tab of the system the Hub shows the translation of sentences with the machine in reference to the parallel sentence for that sentence in the parallel document fed. One example for the Amharic to English System is shown in Figure 5.

Amharic
በተለይም ዳግሞ፡ የተማሪዎቹ ተቃውሞ የዘውግ ፖለቲካን እና የከተማ መረት መቀራመትንም ያዋህዳ መሆኑና የኢትዮጵያ ዳያስፖራዎች በአገር ውስጥ ፖለቲካ ጉዳይ መሳተፋቸው ነገሩ መብራት አንዲያስፈልገው ያደርጋል።

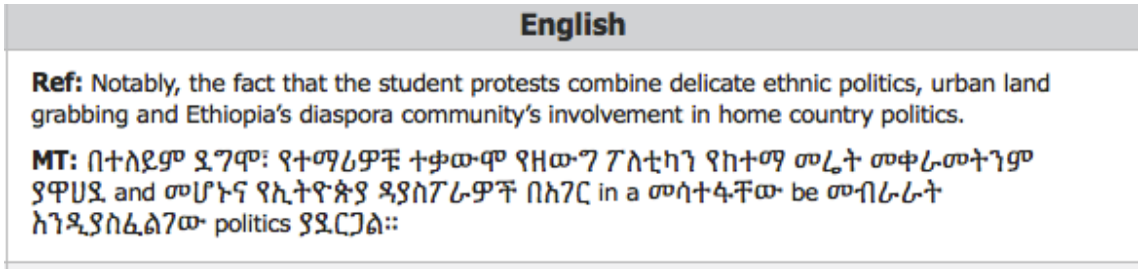


Figure 5 sample taken from "Evaluation" tab of the Amharic-English system

The box with the title Amharic is the Amharic text and the sentence which starts with the word "Ref" in the English box is the translation of the Amharic sentence as it appears in the parallel document fed, and the sentence starting with "MT" is the Machine translation result. As can be seen the system fails to even produce English words or the corresponding Amharic words as they do not appear more frequently in the document.

However in Figure 6 the system managed to produce the translation almost in English, Even Though the meaning is not accurate.

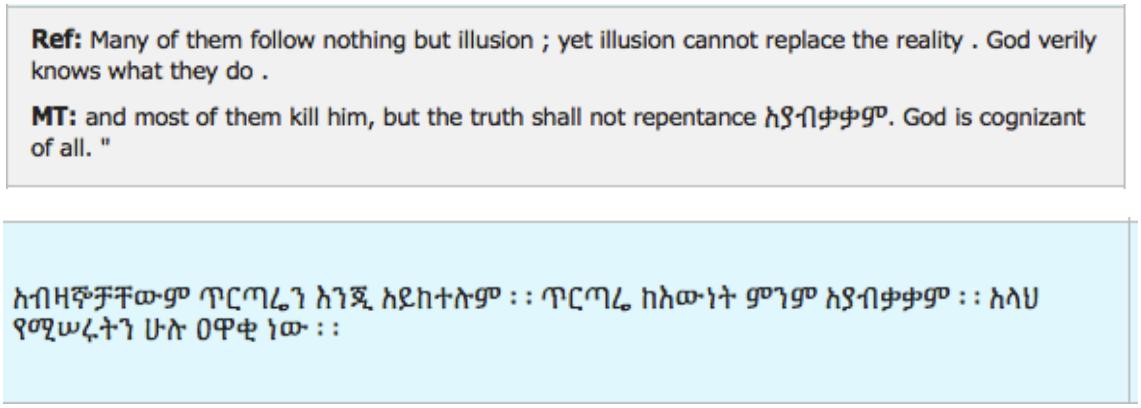


Figure 6 sample taken from "Evaluation" tab of the Amharic-English system

The English German pair systems were set up to see how much the size of the parallel corpora affects the quality of the translation system trained in the Hub. A relatively high amount of data were fed into this systems and there were also more free sources which could be used if a better result is desired. For the English-German System 10,438,686 sentences were extracted and 10,118,467 were aligned. The BLUE score is 29.53 (+3.97). While for the German English system 10,366,592 sentences were extracted and 10,118,467 were aligned. The BLUE score is 35.59 (+3.24). The two systems can produce sentences in the target language in most cases and the translations are relatively accurate.

For the German to Amharic system 36,047 sentences were extracted and 30,860 sentences were aligned. The BLUE score for the system is 11.02. For the Amharic-German system 36,047 sentences were extracted and 30,860 sentences were aligned. The BLUE score is 10.27.

One thing to mention here is a BLUE score cannot be used to compare the quality of two translation systems. For example the BLUE score for Amharic to English system is 7%, while for Amharic to German it is 10.27. But this does not guarantee the first is better than the second. However a high Blue score difference most likely shows quality difference.

4. Implementation of Application

The application was built using Xcode, Swift language, the translator API discussed above and Sketch 3 to design the Icons used in the Application. The codes and the layout were done so that it would be straightforward to add other languages than Amharic and English. After taking a look at the other translate applications in the store, I tried to give the application the functionalities the other applications included except language detection and speaking of text as the Microsoft translator Hub does not support these functionalities yet.

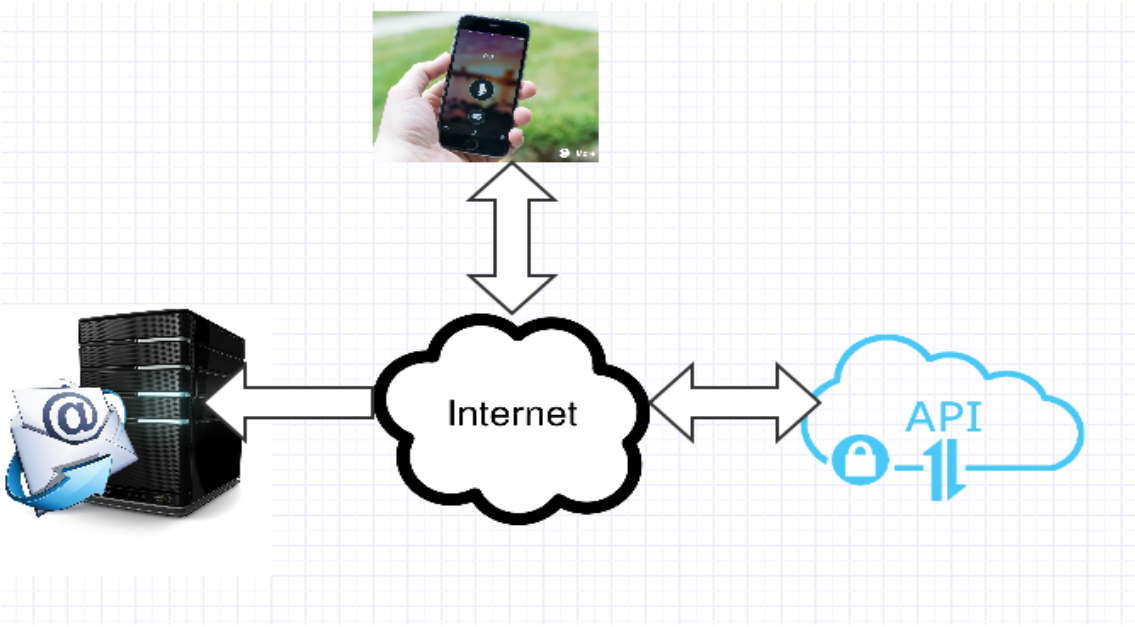


Figure 7 System diagram

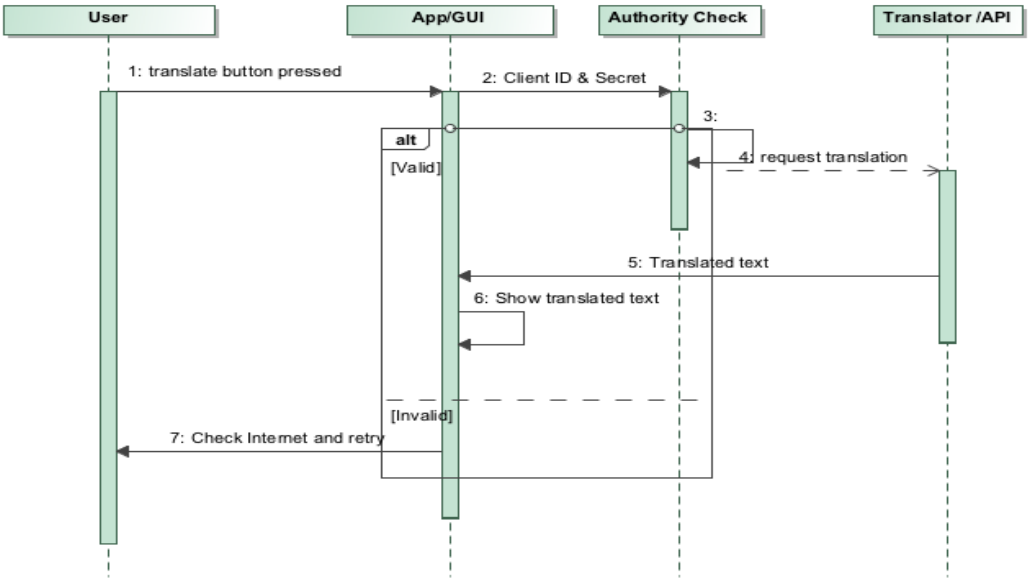


Figure 8 sequence diagram

As shown Figure 7 the application makes use of internet to access the Microsoft translator API and to send a feedback message. The user interaction with the application and the associated operations to get the translated text can be modelled using the sequence diagram in Figure 8. The user enters the text and presses the translate button. Then an HTTP request will be sent to get the access token. If the application is referring to a registered application on the Azure data market, then another HTTP request will be sent to access the translation method. Then the application presents the translated text to the user. Another alternative way is the user does not have an internet connection and will be notified with an error message.

4.1 Designing the application

Before designing the application the structure of the translator applications on the Application store such as Google translate, iTranslate ,iHandy translate was evaluated. The motivation for the project was personal curiosity as to how translation systems work and developing skills for building an application for iPhone, and not building an application which can be competent on the market with the above mentioned applications on the store. After analyzing those applications features which can be implemented with the resources available, the features were selected and designed accordingly.

As can be seen in Figure 9 the application has a Home screen where a user can choose a source and target language for translation and enter a text to be translated. A user can also swap between language pairs. The languages are listed in a TableView whose ImageView is set to the flag of the country where the language is spoken and the name of the language. This will make it straightforward while adding more languages later on. For now both the source and language table views have three

languages, i.e. Amharic, German and English. When a user taps on the source language button or target language button, the corresponding Storyboard will appear and the user can choose the language from the list in the TableView.

When a user taps the TextField to enter the text a toolbar appears. The toolbar contains a button to dismiss the keyboard named “cancel”, a button to clear the text of the TextField called „clear“ and a button to translate the entered text into the target language called “translate“. If the user presses the translate button the translated text appears in the TextField below the source TextField. There is also a speaker to listen to how the sentence is read if it is English text. A user can navigate through the application using NavigationBar and the TabBar The user can navigate to choose the languages using the NavigationController BarButton Items and can navigate to the History and Settings section of the application using a TabBarController.

Tapping the History BarButtonItem on the TabBarController will take the user to the View which contains a TableView which is populated with translation history. As can be seen in the second screen of Figure 9 the TableView contains TableViewCells which show the last translated texts. Each TableViewCell contains two image views to show the source and target language flags and two TextFields to show the source and translated text. A user can delete each individual history by Swiping the TableViewCell to the right and pressing Delete.

The third part of the application is the setting part which contains a button to clear the translation history, and a TableView which contains two cells. The first cell which was titled “Feedback“takes to a MailView which uses the UIMessage framework to send a Feedback. And the second cell takes to a View where one can read about the application and was titled “About”. Tapping the Settings BarBiuttonItem on the TabBarController takes the user to this section of the application.

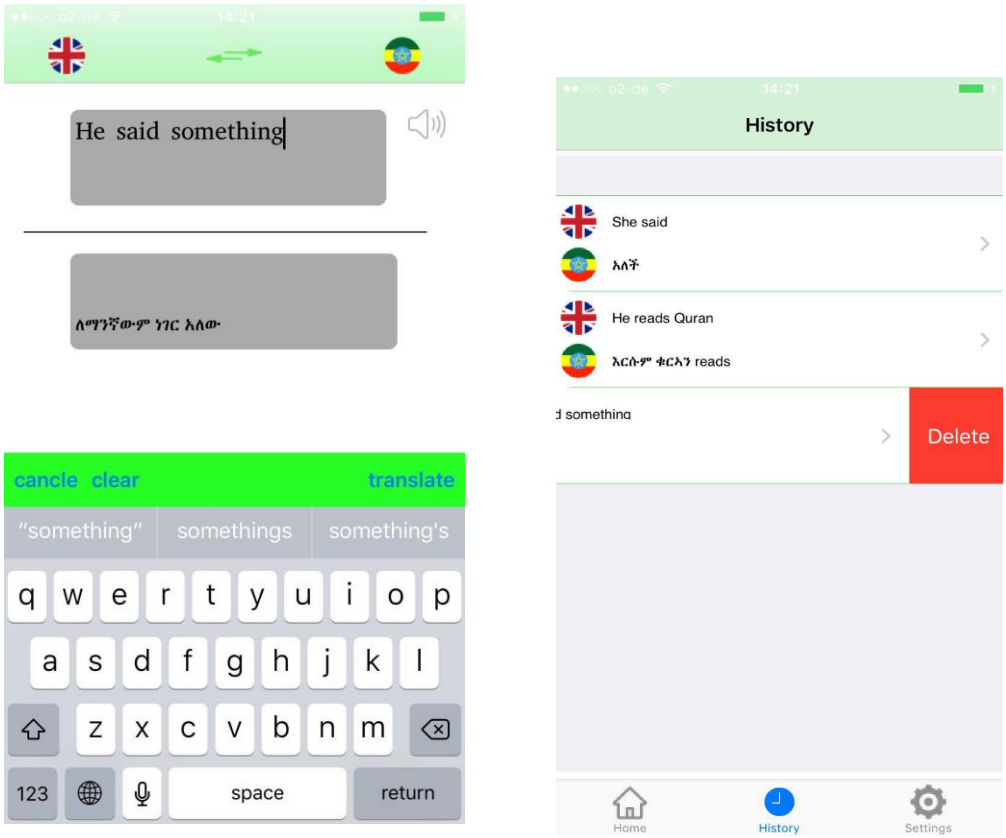


Figure 9. Screen shots of the Home screen and History section

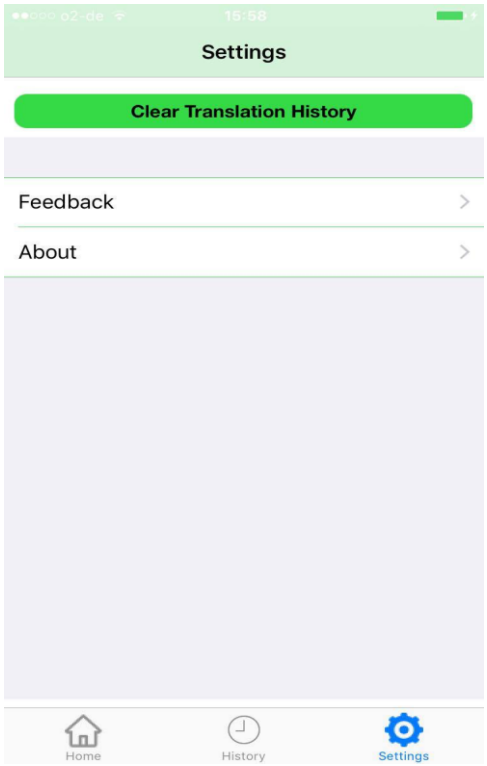


Figure 10 Screen shot of the settings View

4.2 Fetching data

The source of data for the applications is the translation system which was built in the Microsoft translator Hub. Using the system is similar to accessing the normal Microsoft translator API except a unique identifier to the system which is called Category ID must be added. The Microsoft translator API provides a service of translation, and other language-related operations to web or client applications. The custom translation models created by Microsoft translator Hub are deployed on Microsoft servers and can be accessed via the regular Microsoft translator API.

To use the Microsoft translator API, signing up for the Microsoft Translator API on the Windows Azure Marketplace is required. While subscribing to the Azure Marketplace for the translation service there is an option of choosing how many words per month one is planning to access and the cost depends on the number of words. The service is free if the character per month is 2,000,000 characters. Above this character limit Microsoft translator API is a paid service.

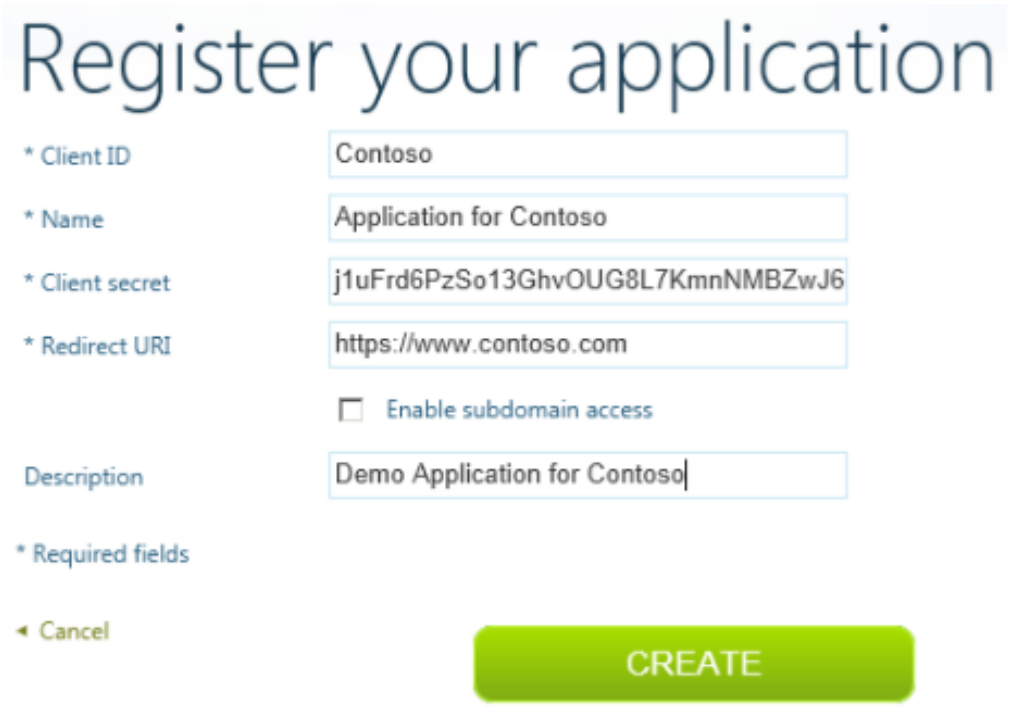


Figure 11 Registering the application to use the Microsoft translator API (Copied from [21])

After registration the next step is to register an application. The client ID field in Figure 11 can be any number or string and the name field is for the name of the application. The Redirect URI is not used by the Microsoft translator API but it is recommended to enter a valid URI for this field. The description is a short summary of the application. The client secret is the most important field. It is generated automatically and it is used together with the client ID when accessing the API from any web or client application which will be using the API. When using a system from Microsoft translator Hub an additional parameter called the category ID is required. It can be found from the Test the Translation page of the deployed project. If no category ID is specified, a generic, Microsoft-supplied translation system will be used. The system which is trained in the Hub has a possibility of choosing from a list of over 7,300 languages to translate. The API supports the following methods.

Methods with category parameter	Methods without category parameter
AddTranslation()	BreakSentences()
AddTranslationArray()	Detect()
GetTranslations()	DetectArray()
GetTranslationsArray()	GetAppIdToken()
Translate()	GetLanguageNames()
TranslateArray()*	GetLanguagesforSpeak()
	GetLanguagesforTranslate()
	Speak()

Figure 12 Microsoft translator API methods (Copied from [21])

All the methods with a category parameter will use the generic translation system for the given language pair if no category is specified. If a category parameter is specific and it is a valid category ID then the specific translation system in the Hub is used. If the category ID specified is not valid then the system fails. The method with no category parameter specified can be used in the same way with both the generic system and the custom system from the Hub. But currently the

GetLanguagesforSpeak(), Speak(), Detect() and DetectArray() cannot be extended to the systems trained in the hub.

Accessing the Microsoft translator API

To Access Microsoft translator API first an access token must be obtained. The access token is passed with each API call and is used to authenticate the access to the Microsoft Translator API. The access token is a way of securing one's access to the Microsoft Translator API .The API will associate the application's requests to the Microsoft Translator service with the associated account on the Azure Marketplace. [9]

After registering the application in the Azure DataMarket an HTTP POST request to the token service can be sent to obtain the access token. The parameters for the token request are URL-encoded and passed in the HTTP request body.

Parameter	Description
client_id	**Required.** The client ID that you specified when you registered your application with Azure DataMarket.
client_secret	**Required.** The client secret value that you obtained when you registered your application with Azure DataMarket.
scope	**Required.** Use the URL http://api.microsofttranslator.com as the scope value for the Microsoft Translator API.
grant_type	**Required.** Use "client_credentials" as the grant_type value for the Microsoft Translator API.

Figure 13 Parameters for getting an access token (Copied from [8])

The response to this request is a JSON contains the access token string which then can be used to access the Microsoft Translator API. In addition to the access token string the JSON returned contains token_type which indicates the format of the access

token and it is Simple Web Token (SWT) for the datamarket. It also contains “expires_in“ which indicates the number of seconds left before the token expires and the scope which shows the domain for which this token is valid. The access token can be used for subsequent API calls before it expires. The token expires every 10 minutes and a new one must be requested every time it expires. There is sample code written in C#, PHP and Windows PowerShell to get the access token in the Microsoft Developer Network (MSDN) website (https://msdn.microsoft.com/en-us/library/hh454950.aspx).The access token will be used as the Authorization header to the calls to the Microsoft Translator API as follows with prefix “Bearer”. "Bearer" + " " + access_token.

A network request in swift can be done using classes NSURL, NSURLRequest and NSURLSession and/or NSURLConnection. (NSURLConnection is deprecated in iOS 9 so the preferred way is to use NSURLSession). To get the access token, an NSMutableURLRequest object was constructed. The method for the HTTP was “POST”.The body and header of the request were set accordingly and the token string was fetched using NSURLSession. From the JSON fetched the access token is the string for the key “access_token” was retrieved using “objectForKey” method of the JSON object. The token was then saved to the NSUserDefaults until it expired. When the token expires, the expired token will be cleared from NSUserDefaults and a new one will be requested.

In the methods used to translate and detect text, first the existence of a valid access token was checked and after that an HTTP request to the translate and detects method of the Microsoft translator API respectively was sent, in the same way the access token was got. The class containing the API access operation has a protocol which was later implemented by the ViewController class.

```
protocol AccessTokenDelegate {

   func translationEroor (ac:AccessTokenRequester, error:String)

   func translatedText(ac:AccessTokenRequester, text:NSString)

   func detectedLanguage(ac:AccessTokenRequester, lan:NSString)

}
```

Listing 1. A protocol whose methods were implemented by the ViewController class

In the ViewController class the three delegate methods shown in Listing 1 were implemented to set the TextField for the translated text with the translation of the text or an error string if there was an error translating the text.

To make representing a language easier a class called LanguageCodes was implement. The LanguageCodes object has properties to represent the name of the language , the flag of the country where the language is spoken (an UIImage object was used to represent this) and the language code as represented by Microsoft Translator API.The properties of the class were defined in Listing 2.

```
var name:String

var langCode:String

var flag:UIImage
```

Listing 2. Properties of the Class LanguageCodes

All the properties which will be used in more than one ViewController class were defined as properties in the AppDelegate class and were accessed by defining an instance of the AppDelegate and accessing its properties.

```
var delegate=UIApplication.sharedApplication().delegate! as!
AppDelegate
```

The following are the properties defined in the AppDelegate

```
var historyArray : [History] = []    - Representing
  an array containing the translation    History object
var languageArray:[LanguageCodes] = []   - To represent all
 the languages used in the application. For now it is only
 Amharic and English
var sourceLan:LanguageCodes! - For the selected source language
var targetLan:LanguageCodes!   - For the selected
         Target language
```

Listing 3 Properties defined in the AppDelegate class

The same way the methods which are described above are implemented. For now only the translate method is implemented since detection and speak are not supported by the hub yet. As described in the section 3.2 the Microsoft translator Hub does not support speak method yet. For this reason the AVFoundation class was used to utter the texts in the English language as shown in Listing 4. This can be accomplished by writing only three lines of code.

```
@IBAction func speak(sender: AnyObject) {
   myUtterance =
            AVSpeechUtterance(string:sourceText.text)
   myUtterance.rate = 0.3
   synth.speakUtterance(myUtterance)
}
```

Listing 4. Implementation of the speak functionality

4.3 History

In the history section, the application saves the last few translations in a persistent storage and shows them in a TableView when the application loads next time. There are many ways to implement persistency in iOS applications such as NSUserDefaults, NSCoding and when saving a large amount of data, CoreData and SQLite database are used. CoreData has an advantage over SQLite that one need not write SQLite statements and it is not needed to have all the instances of the class saved in CoreData in memory inorder to access them and it is faster when querying.

```
class History: NSManagedObject {
                    @NSManaged var sourceText: String
                    @NSManaged var targetText: String
                    @NSManaged var sourceLan: String
```

```
                    @NSManaged var targetLan: String
}
```

Listing 5 Class to defined fields to be saved into CoreData

As shown in Listing 5 a class called History was created to handle the data saving in CoreData. sourceText is to save the text in source TextField and targetText to save the text in the target TextField and sourceLan to save the name of the source language and targetLan for name of the target language.

When a user presses the translate button it will also be saved to CoreData and populated in the History section of the application when the view appears.The TableView is made of custom cells with two ImageViews for the flags of the source and target languages and two TextViews for the source and target texts. Since it is advisable not to save Images in CoreData if possible, the name of the language was saved and a function which retrieves the image from the name of the language was used.

```
func getLanCode(name:String)->LanguageCodes {

    var langCode:LanguageCodes!


for(vari=0;i<self.langSourcedelegate.languageArray.count;i++){

      if(self.langSourcedelegate.languageArray[i].getLangName()
==name){

                    langCode =
self.langSourcedelegate.languageArray[i]

                    break

       }
```

```
    }
   return langCode
}
```

Listing 6 Function to get the LanguageCodes object from a string name of language

4.4 FeedBack Section

The Message UI Framework can be used to present standard composition interfaces for email and SMS (Short Messaging Service) text messages with the use of specialised ViewControllers without leaving the application.[11].The mail ViewController is presented modally and the user has options of customising the message before sending it or cancelling the message. The Message UI Framework has three classes: a UINavigationController, MFMailComposeViewController which is used to compose a mail and MFMessage ComposeViewController to compose SMS messages. In the project MFMailComposeViewController is used for the user to send his/her feedback to the email address provided.

5. Amharic Custom iOS Keyboard

The Amharic writing system is based on a version of the Ge'ez script known as ፊደል (Fidel). [17] The earliest known inscriptions in the Ge'ez script date to the 5th century BC. The writing direction is left to right in horizontal lines. Each symbol represents a syllable consisting of a consonant plus a vowel which is different from English. For example the Amharic letter “በ” represents “be” in English which is a combination of one vowel and one consonant. The different vowel and consonant nominations are represented as different letters.

Table 1. The first Amharic letter with its variations and corresponding pronunciation in English

ፊደል	English
ሀ	Ha
ሁ	Hu
ሂ	Hi
ሃ	Ha
ሄ	He
ህ	Hei
ሆ	Ho

However, there is no standard way of translating the Amharic alphabet into Latin and some sounds are not even representable by Latin letters. The ፊደል system has also its own numbers, but the numbers are not often used in modern writings with the language and are not taught well in school compared to the Arabic number system. Therefore number-related writings use the Arabic numbers.

A custom keyboard is an application extension supported by iOS to program one's own keyboard which is integrated in the application and it is particularly useful when programming a keyboard not supported by iOS. When switching to a custom keyboard it replaces the system keyboard and responds to taps, gestures, or other input events and provides text, in the form of an unattributed NSString object and inserts the text into the input field such as TextView and TextField. The minimum requirement for a custom keyboard is that it has the next keyboard key which makes the user able to switch to the system keyboard or other keyboards installed in the device.

Once a user chooses the custom keyboard it will be the keyboard for all applications the user opens and it is necessary to have the ability to switch to the system keyboard. In the system keyboard this next key is represented with the globe icon. The custom keyboard does not have access to most of the keyboard settings the system keyboard has, such as Auto-Capitalization and Enable Caps Lock. If it is needed that the custom keyboard should have this functionality, one should create one's own standard settings bundle.

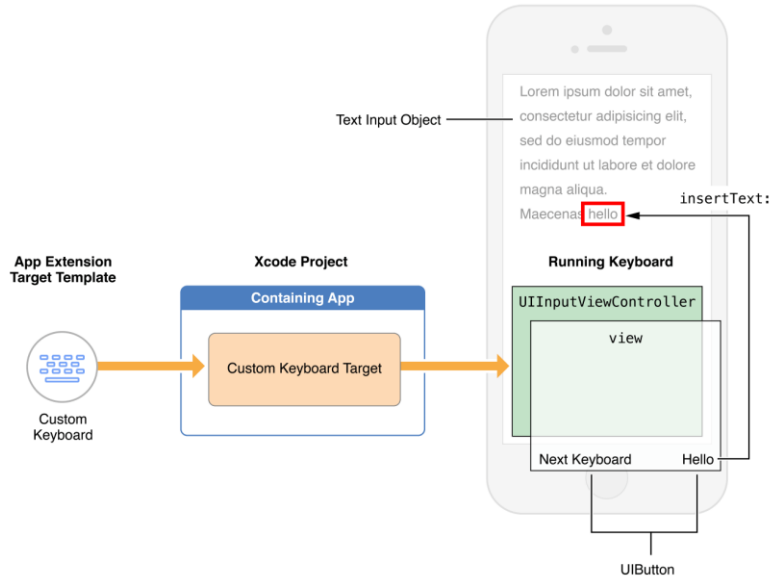


Figure 14 Basic structure of a custom keyboard (copied from [21])

The iOS custom keyboard template contains a ViewController which is a subclass of UIInputViewController with the basic implementation of the required next keyboard key. For all the application extensions there is no window in the target and therefore, no root view controller and the default ViewController is the primary ViewController. In addition to the ViewController the template contains an Info.plist file preconfigured with the minimal values needed for a keyboard.

As discussed in the introduction section the Amharic language has more than 250 letters and numbers and the font is not supported by iOS. For this reason I have included the Abyssinica SIL font which supports around 40 languages in Ethiopia and Eritrea. The .ttf (True Type Font) file was included in the custom keyboard directory and added to the info.plist file as the fonts provided by the application. The most useful and common Amharic letters are 33 and they have 7 variations i.e. the first letter of Amharic is “ሀ” and its variations are { "ሀ","ሁ","ሂ","ሃ","ሄ","ህ","ሆ"} . It is obvious that it is not possible to represent them on one page of the smartphone keyboard space. In the

iOS system keyboard this is accomplished by popping a view containing the extra keys even above the top row, but this functionality is not possible with the custom keyboard.

For this reason the height of the custom keyboard was increased to leave more space for the extra keys. This made the keys relatively small, but it is the easiest way to solve the problem.

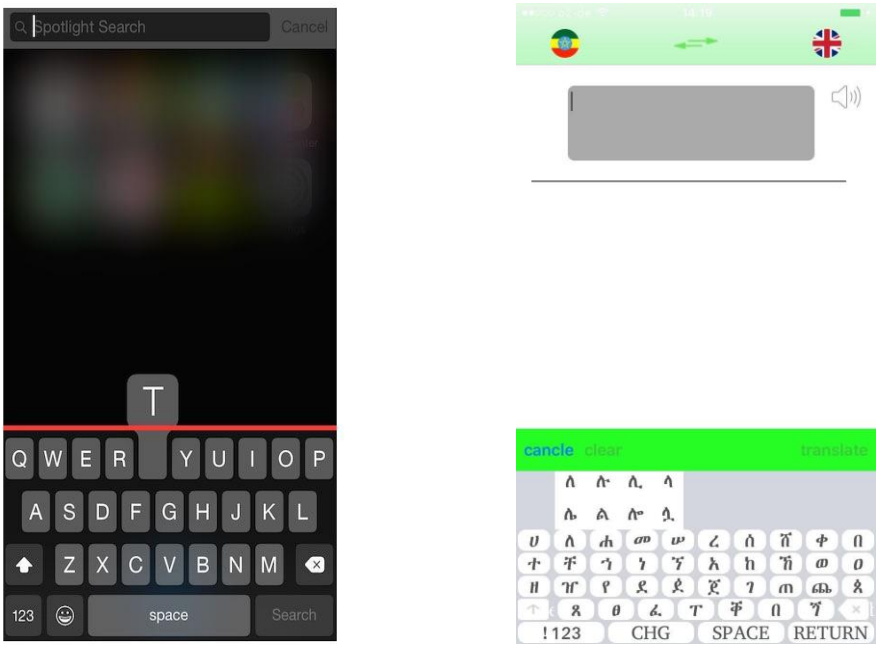


Figure 15 Custom Keyboard

AS can be seen from the figure 15 the 37 first letters are represented in the first page of the keyboard with the up arrow key to show a view containing other sets of keys with

“123” keys for the view with Amharic numbers and punctuation marks. The next keyboard key is represented as “CHG” and a space and return key. Long pressing the first letter key will pop a view containing the variations of the key, in most cases they are 8 but the most common ones are 7 for each first letter.

To implement the popping keys the UILongPressGestureRecognizer class was used. When a user taps a key for more than the time set, the extended View for the Key will appear. The user can then choose from the popping view the letter he/she wants to type. To implement this all the letters with variations are represented in a dictionary ({String: [Strings]}) by making the first key the key and the other keys values. This made it easier not to implement the LongPressGesture to the keys which do not need the Extended View.The extended views contain 7-9 keys to represent the variations keys which are related to the given key, even though the most commonly used are 7. Other keyboards on the store such as Amh Keyboard have represented these extra keys and are also in the list of the Unicode character for the Amharic alphabet.

The popping views pop from left to right if the key is to the left of the middle of the screen and from right to left if the key is to the right of the middle of the screen. Therefor there is no hidden part of the popping view when the key approaches the end of the keyboard main view.

6. Comparison of Translators

There are a lot of translator software on the store. Google translate, Microsoft translator, iTranslate and iHandy translator among them. Google translate as for March 2016 supports 103 languages.

6.1 Google Translate

Google translate allows four ways of entering a text for translation. One can type the text or draw the text with finger tips (if for example the keyboard is not supported in the system), or speak the word into the microphone and it will be converted into text for translation and lastly document translation i.e. one can take a picture containing written text and the translator will identify the text in the document and give the translation for the identified text. Another interesting feature of google translate is the live conversation feature. Live conversation is a feature where the speech out of two languages is detected and translated accordingly. This is quite useful when verbally communicating with someone who speaks the other language, it helps through the conversation. The offline mode is another beneficial feature of Google translate. To use the offline mode, the pack for the language needed should be downloaded, though for some of the features and in some of the languages this feature is not supported. [12] Additionally Google translator has the feature to pine a translation and synchronize them across devices and access them in any way one wants. It also suggests the language if user enters the wrong language source and text combination i.e. if you type an English sentence choosing Finnish as the source language, the translator suggests to switch the language into English. Another feature is to reverse translate a sentence. For example if one has a sentence in English and its translation in Amharic, one can give the output Amharic text as input. Translating it does not give the same result in most cases. This implies the translator is way too far from accurate and can be used to evaluate the translator and also for further improving it. Google translation has also a

way to translate a text from message application with no need to copy directly and can also be used as a dictionary. If one enters a single word for translations it gives a list of translations which is the same as one would get using normal dictionary. Google translate has applications for most of the mobile devices in the market and also web-based translation.

6.2 Microsoft Translate

Microsoft translator supports 53 languages as of March 2016. It supports both speech and text translation. Microsoft translate has a good looking user interface and also has features such as text translation from an image, pinning a translation, saving translation for later use called translation history. [13]

6.3 iTranslate

iTranslate supports over 90 languages. iTranslater has a feature to share your translations over social media, mail or message. The free version of iTranslate has only features such as text translation, saving the history for later use and hearing how the word is pronounced. The free version of iTranslate does not support speech translation and also it has a limitation to the length of the sentence translated and there are ads popping in the free version. However in the premium version one can translate any length of text, speech translate and there are no ads showing [13]

6.4 iHandy

iHandy translate supports 52 languages and allows only text translation, saving the translations for later use and sharing the translation over Facebook, Mail and Message applications. But in the pro version speaking out the text, speech translation,

translating an unlimited length of the sentence, and free of add translation is supported. [13]

6.5 Comparison

In terms of supported languages google translate takes the lead by supporting 103 languages. Microsoft translate has relatively good looking user interface compared to the above discussed translators. Google and Microsoft translators are free of charge for all kinds of supported functionalities and there are no ads popping while using the translator. However one should pay $5 to use the premium version of iTranslate and $1.99 to use the Pro version of iHandy translate. The live conversation of Google translator and the reverse translation are features which are not supported by any of the other translators. Translating an image, writing text with finger tips and offline translation are currently supported by only Google and Microsoft translate. Overall Google translate has a lot of features which are helpful while translation foreign language. Most of the features mentioned while discussing the Google translate and not mentioned in the section for the other ones are features supported only by Google translate. When it comes to accuracy it is difficult to say one is better but translators with a dictionary feature such as Google and iTransalte are good when translating words. And most of the translators can translate short sentences well but fail while translating long sentences.

6.6 Amharic Translators

Since February 17, 2016 Google translate supports the Amharic language and it is the only general purpose Amharic translator on the Application store so far. Currently Google supports only text translation for the Amharic language. I.e. features such as uttering the Amharic text, speaking in Amharic and translating an Amharic sentence from a picture are not supported yet. Google also provides with the text written in

English while translating i.e. the translation of the word “Hello” to Amharic is provided in the Amharic alphabet plus how it would be pronounced in English letters, which is “selam”. The accuracy of the translation for Amharic is very low for Google translate compared to the translation for example from English to German. This is Google translator’s problem for all under - resourced languages.

Compared to the application done in this project Google translate gives better results though not satisfactory. The reason for this is that both the Google translator and the translation system in this project are based on the same underlying principle which is statistical machine translation. As discussed many times in this paper this system is based on millions of parallel documents and the accuracy of the system mainly depends on the amount of data fed. At this point using Google translate API will probably give a better result, Even though the Google API is a paid service. One advantage of the application is one can read the Amharic text within the application without installing any Amharic reader software, but using Google translate the software should be installed for the iPhone device to read the text.

7. Results and Discussion

The application was tested in an iPhone 6 device and iPhone 5 and iPhone 6 simulators and the user interface looks the same in all cases. It also has the same layout for landscape mode. Most of the elements of the UI are aligned relative to each other and their basic layouts are independent of the device size. In the keyboard section the keys are relatively small for the portrait mode and there is a gap between the first row keys and the keyboard toolbar. As mentioned in chapter 5, the reason for this is it is not allowed to pop keys above the top edge of a custom keyboard's primary view. If a better looking keyboard is needed a separate keyboard can be programmed using third party libraries.

The performance of the application in terms of speed is satisfactory, except there is a delay in the first translation while the request is sent to get the access token. This happens approximately every 10 minutes if successively used since the access token needs to be requested every 10 minutes.

The translation quality is low especially for Amharic-English and Amharic-German pairs. In the case of translating long sentences the system even fails to produce words in the target language. As briefly mentioned in the chapter 3 the reason is the parallel documents fed are not enough to produce a good quality general purpose translator. Google for example uses millions of big parallel documents such as parallel books though the systems produces are still too far from perfection.

One way to improve the translation system is to make parallel corpora by extracting sentences from websites and other documents and use the free sentence processing tolls on the internet, or use the free TMX editors and make parallel documents sentence by sentence. This method is good if parallel documents which are in a format which is not supported by hub such as Xml is found. The whole process is time taking

but is a straightforward way of making such documents for someone who has no knowledge of parallel document processing.

The Microsoft translator Hub provides a good way to improve the translation system. In the test interface of the deployed project there is a way to add to the correct translation of a word or sentence which is not correctly translated by the system. This is the community translation service which was discussed in section 3.1. In the application the addTranslation () method can be implemented to let users add their contribution to improve the translation quality of the system. For the time being the method is not implemented, but it will soon be implemented.

The English-German systems has an optimal quality even though it was not the main focus of the project. For both the German-English and English-German pairs the BLUE scores are good and they have a plus BLUE score from the first training which shows the score compared to the Microsoft model. The result from this system will be used for possible extension of the project to include more languages.

There were many successive trainings done to both Amharic-English and English-Amharic systems by modifying some of the data sets, but the results were less than one BLUE point score and were not significant enough to have an effect on the translation quality. In most of the successive trainings the BLUE score decreased by approximately 0.1, which is an implication that the training did not produce a better system than the preceding one.

8. Conclusion

The aim of this project was to build a language translator application for the Amharic-English language pair. As a result an iPhone application was made which was later extended to include the German language. The translator has only text translation functionalities. Other language processing functionalities such as speech recognition, language detection are not supported currently, as the Microsoft translator Hub does not support them at the moment.

The overall thesis project was carried out in three steps. First a research was done on how translation systems work, then the Microsoft translator Hub was selected to make a statistical machine translation system. A translation system was built on it. Finally an iPhone application was developed using the Microsoft translator API to access the translation system built on the Microsoft translator Hub.

Even Though the goal of the project was achieved the system does not produce good translation results compared to the corresponding human translations. The system will be continuously improved using a community translation mechanism discussed in chapter 3 and also by adding more documents to the system. The system is under improvement and the application is not yet published to the application store.

8. References

1. Philipp Koehn. Statistical Machine Translation. New York: Cambridge University Press; 2010.

2. Mulu G.Teshome, Laurent Besacier. Preliminary Experiments on English-Amharic Statistical Machine Translation. Addis Ababa: Addis Ababa University, 2012.

3. KantanMT. What is Machine Translation? [online]. Ireland: Xcelerator Machine Translations Ltd, 2015.
URL: https://kantanmt.com/documents/Machine_Translation.pdf
Accessed 10 March 2016.

4. Microsoft research. Translator Hub. Microsoft.
URL:http://research.microsoft.com/enus/projects/microsofttranslatorhub/microsoft_translator_hub_overview.pdf
Accessed 10 March 2016

5. Chris Callison-Burch. Machine translation: Word-based models and the EM algorithm [online]. United States : John Hopkins University; 2007
URL : https://www.cs.jhu.edu/~jason/465/PowerPoint/lect32a-mt-word-based-models.pdf
Accessed 10 March 2016.

6. Aynalem Tesfaye, Kevin Scannell. Amharic – English Cross-lingual Information Retrieval: A Corpus Based Approach. Haramaya: Haramaya University; 2012.

7. Rachel Killackey. Statistical Machine Translation from English to Tuvan* . United States: Swarthmore College; 2013.

8. Microsoft Developer Network. Obtaining an Access Token [online].Microsoft. URL: Microsoft https://msdn.microsoft.com/en-us/library/hh454950.aspx Accessed 10 March 2016

9. Microsoft Research. Machine Translation [online]. Microsoft URL: http://research.microsoft.com/en-us/projects/mt/ Accessed 10 March 2016.

10. Microsoft Developer Network. Translator Language Codes [online]. Microsoft URL: https://msdn.microsoft.com/en-us/library/hh456380.aspx Accessed 10.3.2016.

11. iOS Developer Library. Message UI Framework Reference [online]. Apple URL:https://developer.apple.com/library/ios/documentation/MessageUI/Reference/MessageUI_Framework_Reference/ Accessed 10 March 2016.

12. Rita El Khoury. Microsoft Translator And Google Translate Compared: Is There A New Challenger In The House? [Online]. Android police, 2015. URL: http://www.androidpolice.com/2015/08/09/microsoft-translator-and-google- translate-compared-is-there-a-new-challenger-in-the-house/ Accessed 10 March 2016.

13. EMily Schiola. Top 5 Best Translator Applications [online]. Heavy, 2015. URL: http://heavy.com/tech/2015/04/best-translation-translator-application-voice- iphone-android/ Accessed 10 March 2016.

14. Kenji Yamada and Kevin Knight. A Syntax-based Statistical Translation Model [online]. California: University of Southern California; 2010 URL: http://www.aclweb.org/anthology/P01-1067 Accessed 10 March 2016.

15. Amittai E. Axelrod. Factored Language Models for Statistical Machine Translation [online]. UK: University of Edinburgh; 2006.
URL: http://homepages.inf.ed.ac.uk/miles/phd-projects/axelrod.pdf
Accessed 10 March 2016.

16. Philipp Koehn. Syntax-Based Models [online]. UK: University of Edinburgh; 2012.
URL: http://homepages.inf.ed.ac.uk/pkoehn/publications/esslli-slides-day5.pdf
Accessed 10 March 2016.

17. Omniglot. Amharic (አማርኛ) [online]. 2016
URL: http://www.omniglot.com/writing/amharic.htm
Accessed 10 March 2016.

18. Andreas Eisele , Christian Federmann , Hans Uszkoreit , Herv´e Saint-Amand, Martin Kay , Michael Jellinghau , Sabine Hunsicker , Teresa Herrmann & Yu Chen. Hybrid Architectures for Multi-Engine Machine Translation. Saarbru¨cken, Germany : Saarland University; 2012

19. Microsoft research. Microsoft Translator Hub User Guide. Microsoft Corporation; October 2015.

20. Microsoft research. Microsoft Translator Hub API Guide. Microsoft.

21. Application Extension Programming Guide. Custom Keyboard. Apple.

Appendix 1: Sample of Phrase Extraction Algorithm for SM

```
Input: word alignment A for sentence pair (e, f)
Output: set of phrase pairs BP
 1: for e_start = 1 ... length(e) do
 2:   for e_end = e_start ... length(e) do
 3:     // find the minimally matching foreign phrase
 4:     (f_start, f_end) = ( length(f), 0 )
 5:     for all (e,f) ∈ A do
 6:       if e_start ≤ e ≤ e_end then
 7:         f_start = min( f, f_start )
 8:         f_end = max( f, f_end )
 9:       end if
10:     end for
11:     add extract(f_start, f_end, e_start, e_end) to set BP
12:   end for
13: end for
function extract(f_start, f_end, e_start, e_end)
 1: return {} if f_end == 0 // check if at least one alignment point
 2: // check if alignment points violate consistency
 3: for all (e,f) ∈ A do
 4:   return {} if e < e_start or e > e_end
 5: end for
 6: // add pharse pairs (incl. additional unaligned f)
 7: E = {}
 8: f_s = f_start
 9: repeat
10:   f_e = f_end
11:   repeat
12:     add phrase pair (e_start .. e_end, f_s .. f_e) to set E
13:     f_e++
14:   until f_e aligned
15:   f_s--
16: until f_s aligned
17: return E
```