The brief view on Google Translate Machine

Omid Karami, Student of Vienna University of Technology

Abstract—I have tried to describe briefly about Google machine translate history and some of the methods are used in it. In the following paper, I have described RbMT and SMT strategy, which are two major Machine translate technique. One of modeling translate is presented in naïve way and also two decoding algorithms. I have also mentioned some basic formulas which are used widely on statistical method.

Index Terms— Google MT, Machine translate, MT, RbMT, SMT.

I. INTRODUCTION

He Machine translation (MT) is automated translation. It is the process by which computer software is used to translate a text from one natural language (like English) to another (like Germany).

To process any translation, human or automated, the meaning of a text in the original (source) language must be fully restored in the target language. However this seems straightforward, it is really complex. Translation is not a just word-for-word substitution. A translator must interpret and analyze all of the elements in the text and know how each word may influence another. This requires extensive expertise in grammar, syntax (sentence structure), semantics (meanings), etc., in the source and target languages, as well as familiarity with each local region.

The challenge between Human and machine translation is to improve quality of machine translation to produce publishable quality translations, quality translations.

There are three common machine translation technologies in commercial use today:

A. Rule-Based MT

Rule-Based Machine Translation (RbMT) systems use large collections of rules, manually developed over time by human experts, which map structures from the source to the target language.

The software parses text and creates a transitional representation from which the text in the target language is generated. This process requires extensive lexicons with morphological, syntactic, and semantic information, and large sets of rules. The software uses these complex rule sets and then transfers the grammatical structure of the source language into the target language.

Translations are built on gigantic dictionaries and sophisticated linguistic rules. Users can improve the out-of-the-box translation quality by adding their terminology into the translation process. They create user-defined dictionaries which override the system’s default settings.

In most cases, there are two steps: an initial investment that significantly increases the quality at a limited cost, and an ongoing investment to increase quality incrementally. While rule-based MT brings companies to the quality threshold and beyond, the quality improvement process may be long and expensive.

B. Statistical MT

Statistical Machine Translation (SMT) systems use computer algorithms that explore millions of possible ways of putting smaller pieces of text together, in an effort to produce a translation that looks best. It utilizes statistical translation models whose parameters stem from the analysis of monolingual and bilingual corpora. Building statistical translation models is a quick process, but the technology relies heavily on existing multilingual corpora. A minimum of 2 million words for a specific domain and even more for general language are required. Theoretically it is possible to reach the quality threshold but most companies do not have such large amounts of existing multilingual corpora to build the necessary translation models. Additionally, statistical machine translation is CPU intensive and requires an extensive hardware configuration to run.
translation models for average performance levels.

C. Hybrid MT

In order to address both quality and time-to-market limitations, many rule-based machine translation developers are augmenting their core technology with statistical machine translation technology in what is referred to as ‘Hybrid’ machine translation.

Rule-based MT provides good out-of-domain quality and is by nature predictable. Dictionary-based customization guarantees improved quality and compliance with corporate terminology. But translation results may lack the fluency readers expect. In terms of investment, the customization cycle needed to reach the quality threshold can be long and costly. The performance is high even on standard hardware.

Statistical MT provides good quality when large and qualified corpora are available. The translation is fluent, meaning it reads well and therefore meets user expectations. However, the translation is neither predictable nor consistent. Training from good corpora is automated and cheaper. But training on general language corpora, meaning text other than the specified domain, is poor. Furthermore, statistical MT requires significant hardware to build and manage large translation models.

II. GOOGLE TRANSLATE MACHINE

Google Translate (GT) is a popular translation service provided by Google to translate a word, a phrase, a section of text or an entire web page into one of 51 languages mentioned below. Google translator cannot only translate words and sentences, but also translate pages, books, and even an entire website. The stated goal of Google Translate is to make information universally accessible and useful, regardless of the language in which it's written. When Google Translate generates a translation, it looks for patterns in hundreds of millions of documents to help decide on the best translation. By detecting patterns in documents that have already been translated by human translators, Google Translate can make intelligent guesses as to what an appropriate translation should be. This process of seeking patterns in large amounts of text is called "statistical machine translation". Since the translations are generated by machines, not all translation will be perfect. The more human-translated documents that Google Translate can analyze in a specific language, the better the translation quality will be. This is why translation accuracy will sometimes vary across languages [1].

The history of Google translate machine start from 2001 based on rule-based MT at first it contain just six language English, France, German, Italian, Portuguese and Spanish. (English to others). Then from 2004 Chinese, Japanese and Korean are added. In development time since 2006 Google is decided to Statistic MT and start to add new languages Arabic and Russian by this model. Furthermore since 2006, Google Translate has used proprietary, in-house technology based on statistical machine translation instead. Since 2007, from the result of SMT molding decided to replace all of rule-based engine with statistic version and now all of language use SMT.

The core algorithm to make 'Google Translate - Machine Translation' works is statistical machine translation (SMT). SMT uses statistical model to determine the word translation. This basic method doesn't follow any language translation rules. To make statistical model, we need bilingual text corpora/corpus. Bilingual text corpus is a database of source sentences and target sentences. For example if we want to build statistical model for English to Spain translation, we need a database of English sentences and Spain translated sentences. The more sentences the better statistical model we have.

Computer will be trained to calculate probability word distribution statistic from above sentences. For example if word AAA has probability 80% to be translated into BBB, then we confident that AAA can be translated into BBB. Since it doesn't rely on any linguistic rule, SMT can be used to make translation any pair languages. Although it need times to make bilingual language corpora, but the result is much better than ruled-based translation.

Figure 1 - basic of creating SMT language model
From Figure 1, first step is collecting many documents from many sources. Then system will align sentences and create database of pair sentences (bilingual text corpus). System will be trained using that corpus. It will analyze the statistic of word distribution in each sentence. The output of this training is language model. Each pair translation has their own language model. Language model will be updated each time the system learn new corpus. Using this language model we can translate other sentences.

A. Bilingual text corpus

We know that Google Translate supports many pair language translations. Google gathers bilingual text corpus from many documents. They scan the original version books and the translated version. They crawl websites which have two or more language versions. Sometimes they hire translators to translate from one language to other language.

After they have bilingual documents, Google do word alignment. They have software that can align source sentences and translated sentences. This software creates database pair of source sentences and translated sentences.

B. Benefit of SMT

SMT have benefits over traditional translation method (e.g: rule based translations):

- Generally SMT translator is not tailored to support specific languages. It builds to support many pair of languages so SMT have better use of resources. It means building SMT translator is cheaper than traditional method.
- Depending on the number of bilingual of corpus, SMT translator gives more natural translations. The more bilingual corpus it has, the more translator trained with new bilingual corpus, the more natural translation it has.

While there are many machine translation software on the internet, Google translator is clearly in the front of the pack. One of Google automatic translator’s clear advantages is the phonetic typing.

Google translator allow user to translate more than just Latin based languages by enabling a web based phonetic keyboard right on the translator. Many languages such as Russian, Greek, Hindu, Serbian, Arabic, and Urdu, have different words other than English, but their words may sound like certain terms in English.

III. STATISTICAL MACHINE TRANSLATION

Statistical machine translation is based on a channel model. Given a sentence $T$ in one language (German) to be translated into another language (English), it considers $T$ as the target of a communication channel, and its translation $S$ as the source of the channel. Hence the machine translation task becomes to recover the source from the target. Basically every English sentence is a possible source for a German target sentence. If we assign a probability $P(S|T)$ to each pair of sentences $(S, T)$, then the problem of translation is to find the source $S$ for a given target $T$, such that $P(S|T)$ is the maximum. According to Bayes rule,

$$p(S \mid T) = \frac{P(S)P(T \mid S)}{P(T)} \quad (1)$$

Since the denominator is independent of $S$,

$$\hat{S} = \arg \max_s P(S)P(T \mid S) \quad (2)$$

Therefore a statistical machine translation system must deal with the following three problems:

- Modeling Problem: How to depict the process of generating a sentence in a source language, and the process used by a channel to generate a target sentence upon receiving a source sentence? The former is the problem of language modeling, and the latter is the problem of translation modeling. They provide a framework for calculating $P(S)$ and $P(T|S)$ in (2) [4].
- Learning Problem: Given a statistical language model $P(S)$ and a statistical translation model $P(T|S)$, how to estimate the parameters in these models from a bilingual corpus of sentences?
- Decoding Problem: With a fully specified (framework and parameters) language and translation model, given a target sentence $T$, how to efficiently search for the source sentence that satisfies (2).
Some of the most important modeling and learning issues are used in a statistical machine translate like Google Translate mentioned as follow: it starts with basic Probability and continue with sums and products, the noisy channel, Bayesian Reasoning, word Reordering, word choice, language modeling, N-grams, Smoothing, Evaluating models, Perplexity, log probability arithmetic, translation modeling, translation as string rewriting, model 2 (language model), model 3, models parameters, word to word alignments, estimating parameter values for w-t-w alignments, bootstrapping, all passible alignments, collecting fractional counts, alignment probabilities, decoding, efficient model training and some so on, in this paper I have chosen a few of these methods and tried to describe them in naïve way.

A. Model 2
At this model, it receives a source English sentence $e = e_1, e_2, ..., e_l$, the channel generates a German sentence $g = g_1, g_2, ..., g_m$ at the target end in the following way:

1. With a distribution $P(m|e)$, randomly choose the length $m$ of the German translation $g$. In model 2, the distribution is independent of $m$ and $e$:

$$ P(m|e) = \epsilon \quad (3) $$

where $\epsilon$ is a small, fixed number.

2. For each position $i$ ($0 < i \leq m$) in $g$, find the corresponding position $a_i$ in $e$ according to an alignment distribution

$$ P(a_i|i, a_i^{-1}, m, e). $$

In model 2, the distribution only depends on $i$, $a_i$, and the length of the English and German sentences:

$$ P(a_i|i, a_i^{-1}, m, e) = a_i(a_i | i, m, l) \quad (4) $$

3. Generate the word $g_i$ at the position $i$ of the German sentence from the English word $e_{a_i}$ at the aligned position $a_i$ of $g_i$, according to a translation distribution

$$ P(g_i | a_i^m, g_i^{-1}, e) = t(g_i | e_{a_i}). $$

The distribution here only depends on $g_i$ and $e_{a_i}$.

Therefore, $P(g | e)$ is the sum of the probabilities of generating $g$ from $e$ over all possible alignments $A$, in which the position $i$ in the target sentence $g$ is aligned to the position $a_i$ in the source sentence $e$ [4]:

$$ P(g | e) = \epsilon \sum_{a_i=0}^{l} \sum_{m=0}^{l} \prod_{j=1}^{m} t(g_i|e_{a_i}) a(a_i | a_j | j, l, m) $$

$$ = \epsilon \prod_{j=1}^{m} \sum_{i=0}^{l} t(g_i|e_i) a(i | j, l, m) \quad (5) $$

IV. DECODING
Decoding algorithm in statistical machine translation is a crucial part. Its performance directly affects the quality and efficiency of translation. Without a good and efficient decoding algorithm, a statistical machine translation system may miss the best translation of an input sentence even if it is perfectly predicted by the model.

A. Stack decoders
Stack decoders are widely used in speech recognition systems. The basic algorithm can be described as following:

1) Initialize the stack with a null hypothesis.
2) Pop the hypothesis with the highest score off the stack, name it as current-hypothesis.
3) If current-hypothesis is a complete sentence, output it and terminate.
4) Extend current-hypothesis by appending a word in the lexicon to its end. Compute the score of the new hypothesis and insert it into the stack. Do this for all the words in the lexicon.
5) Go to (2).

B. Scoring the hypotheses
In stack search for statistical machine translation, a Hypothesis $H$ includes (a) the length $l$ of the source sentence, and (b) the prefix words in the sentence. Thus a hypothesis can be written as $H = l : e_1 e_2 ... e_k$, which postulates a source sentence of length $l$ and its first $k$ words. The score of $H$, $f_H$, consists of two parts: the prefix score $g_H$ for $e_1 e_2 ... e_k$, and the heuristic score $h_H$ for the part $e_{k+1} e_{k+2} ... e_l$ that is yet to be appended to $H$ to complete the sentence. From (3) can be used to assess a hypothesis. Although it was obtained from the alignment model, each word $e_i$ in the hypothesis contributes the probability of the target sentence word. For each hypothesis, we use $SH(j)$ to denote the probability mass for the target word $g_j$ contributed by the words in the hypothesis:
To guarantee an optimal search result, the heuristic function must be an upper-bound of the score for all possible extensions \( e_{k+1}e_{k+2}\ldots e_i \) of a hypothesis. In other words, the benefit of extending a hypothesis should never be underestimated. Otherwise the search algorithm will conclude prematurely with a non-optimal hypothesis.

On the other hand, if the heuristic function overestimates the merit of extending a hypothesis too much, the search algorithm will waste a huge amount of time after it hits a correct result to safeguard the optimality.

Due to physical space limitation, we cannot keep all hypotheses alive. There is a possibility to set a constant \( M \), and whenever the number of hypotheses exceeds \( M \), the algorithm will prune the hypotheses with the lowest scores. In an experiments the authors decided to set \( M = 20,000 \).

There was time limitation too. It was of little practical interest to keep a seemingly endless search alive too long.

Since the heuristic function overestimates the merit of extending a hypothesis, the decoder always prefers hypotheses of a long sentence, which have a better chance to maximize the likelihood of the target words. The decoder will extend the hypothesis with large \( I \) first, and their children will soon occupy the stack and push the hypotheses of a shorter source sentence out of the stack. If the source sentence is a short one, the decoder will never be able to find it, for the hypotheses leading to it have been pruned permanently.

V. PHRASE TRANSITION

The phrase translation model is based on the noisy channel model. We use Bayes rule to reformulate the translation probability for translating a foreign sentence into English \( e \) as

\[
S_H (j) = \varepsilon \sum_{i=0}^{k} t(g_i \mid e_i) a(i \mid j, l, m) 
\]

Each foreign phrase \( g \) in \( g'_1 \) is translated into an English phrase \( e_i \). The English phrases may be reordered.

Phrase translation is modeled by a probability distribution \( \varphi(g_i \mid e_i) \).

Recall that due to the Bayes rule, the translation direction is inverted from a modeling standpoint.

Reordering of the English output phrases is modeled by a relative distortion probability distribution \( d(a_i - b_i - 1) \), where \( a_i \) denotes the start position of the foreign phrase that was translated into the \( i \)th English phrase, and \( b_i - 1 \) denotes the end position of the foreign phrase translated into the \( (i-1) \)th English phrase.

In all our experiments, the distortion probability distribution \( d(.) \) is trained using a joint probability model. Alternatively, there is also possibility to use a simpler distortion model \( d(a_i - b_i - 1) = a_i^{b_i - 1} \) with an appropriate value for the parameter \( \alpha \).

In order to calibrate the output length, we introduce a factor \( W \) for each generated English word in addition to the trigram language model \( PLM \). This is a simple means to optimize performance. Usually, this factor is larger than 1, biasing longer output.

In summary, the best English output sentence \( g \) best given a foreign input sentence \( g \) best according to our model is:

\[
e_{\text{best}} = \arg \max_e \ p(e \mid g) \]

\[
e_{\text{best}} = \arg \max_e \ p(g \mid e) p_{LM} (e) w^{\text{length}(e)}
\]

where \( p(g \mid e) \) is decomposed into

\[
p(g'_1 \mid e'_1) = \prod_{i=1}^{k} \varphi(g_i \mid e_i) d(a_i - b_{i-1})
\]

VI. PHRASE DECODER

The phrase-based decoder employs a beam search algorithm, similar to the one by [6]. The English output sentence is generated left to right in form of partial translations (or hypotheses).

It starts with an initial empty hypothesis. A new hypothesis is expanded from an existing hypothesis by the translation of a phrase as follows: A sequence of untranslated foreign words and a possible English phrase translation for them is selected. The English phrase is attached to the existing English output sequence. The foreign words are marked as translated and the probability cost of the hypothesis is updated.
The cheapest (highest probability) final hypothesis with no untranslated foreign words is the output of the search. The hypotheses are stored in stacks. The stack $s_m$ contains all hypotheses in which $m$ foreign words have been translated. Then recombine search hypotheses as done by [7]. While this reduces the number of hypotheses stored in each stack somewhat, stack size is exponential with respect to input sentence length. This makes an exhaustive search impractical.

Thus, we prune out weak hypotheses based on the cost they incurred so far and a future cost estimate. For each stack, it can be keep only a beam of the best $n$ hypotheses. Since the future cost estimate is not perfect, this leads to search errors. Our future cost estimate takes into account the estimated phrase translation cost, but not the expected distortion cost.

It computes this estimate as follows: For each possible phrase translation anywhere in the sentence, we multiply its phrase translation probability with the language model probability for the generated English phrase. As language model probability can be use the unigram probability for the first word, the bigram probability for the second, and the trigram probability for all following words.

Given the costs for the translation options, it can compute the estimated future cost for any sequence of consecutive foreign words by dynamic programming.

During translation, future costs for uncovered foreign words can be quickly computed by consulting this table. If a hypothesis has broken sequences of untranslated foreign words, we look up the cost for each sequence and take the product of their costs.

The beam size, e.g. the maximum number of hypotheses in each stack, is fixed to a certain number. The number of translation options is linear with the sentence length. Hence, the time complexity of the beam search is quadratic with sentence length, and linear with the beam size.

VII. CONCLUSION

I have tried to discover topic about Google machine translate, which is contained a large area of research in Artificial intelligence. I consider two major technique which are used to implement an engine of a MT. each technique has its advantages and disadvantages. Regarding Latin’s language or languages which are simple in their linguistic and rules, the rule-base technique is efficient and easier to implement but in such case like Google MT, which is Support large Varity of languages. There is ability to use large amount of data as train data for different language. It’s trivial to use SMT as engine method. Actually In this area Google research are very active and we can see the acceptable result of Google MT. I think in near feature we can higher accuracy in Google MT.

REFERENCES