Tagging and Statistically Translating Latin Sentences

Andrew Runge

June 10, 2010

Abstract

In developing language translation software, an increasingly common method is to tag words based on their role in the sentence in order to determine where they should be in the sentence, and then put them in that slot to create a basic, sometimes awkward translation. The goal of this project is to tag the sentences and then use a new method of statistically analyzing the words based on part of speech pairings in order to generate the most sensible and accurate translation of a Latin sentence. The generated hypotheses will also receive user-feedback to correct any errors and improve the translation quality.

Keywords: Machine Learning, Statistical Translation, N-Gram

1 Introduction

The field of machine translation has been growing significantly over the past few years in order to make computers more helpful and useful in interacting with humans. Language translation has evolved greatly through the use of methods such as word tagging. By tagging words for their important characteristics, it allows the computer to greatly narrow down the range of possible translations. It helps the computer to better recognize the role a word plays in a sentence, and from there helps it make better decisions about where the word should go in the sentence with relation to the other words.

In addition to the use of word tagging, statistical strategies for looking at word placement greatly helps the computer in creating and organizing a sensible sentence. By looking at the components of the sentence based on their position with relation to other words, as well as by comparing the computer's generated hypotheses for translations in comparison to other sentences of the same language, it allows the computer to get a better frame of reference as it develops better and better hypotheses. N-grams, sets of words n words long, are used in the area of statistical translation, as they allow the computer to look at a specific number of words around its target. If the computer were to try to look at each word individually, then it would have little luck in creating a sensible sentence. How-

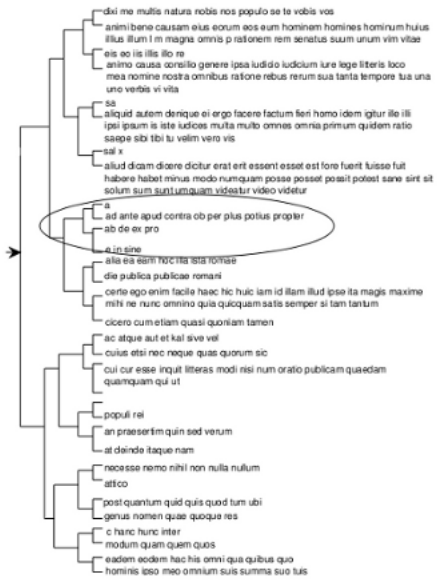

Figure 1: Tree of words sorted by sentence role from the assorted works of Cicero generated by the methods of McMahon and Smith.

ever, by looking at the word's context and other words around it, the computer can determine where the word should go in relation to the rest of the sentence and subsequently create a more grammatically correct and understandable sentence.

By using these two methods, statistical translation and word tagging, in conjunction with each other, the computer can build the best possible knowledge base in order to then go about translating the sentence. In order then to improve upon this base the program can be converted into a Machine-Assisted Translation program. This style of translation outputs the first part of the translation and then seeks confirmation or correction in order to better guarantee a good translation for the second part of the program.

The word tagging gives the computer some information ahead of time about how the word functions in the sentence, as well as allowing it to associate it with other words in the sentence in order to make grammatical sense. At the same time, statistical translation can make use of this information to eliminate even more possible hypotheses, and narrow down its scope so that way it can be more efficient in its analysis of the hypotheses as well as reducing the amount of work the statistical analysis section has in terms of fixing the sentence so that each of the components agree with each other and make sense. Finally, the user-input confirms or corrects the computer's translation, ensuring the best possible final result.

2 Background

The project tested in Two-Stage Hypotheses Generation for Spoken Language Translation used n-grams to generate multiple possible translations for a given sentence and then statistically selected the best one based on a series of tests to "score" each hypothesis. Their method generated understandable and grammatically correct sentences that were largely able to preserve the original meaning of the sentence. This is one of the biggest challenges of language translation, and so their success reinforces the usefulness of n-grams in doing so.

McMahon and Smith also demonstrated use of n-grams in word tagging for part of speech purpose. They generated trees of the

$$h_{IBM}(f_1^J, e_1^I) = \log\left(\frac{1}{(I+1)^J}\prod_{j=1}^{J}\sum_{i=0}^{I} p(f_j|e_i)\right)$$

Figure 2: An equation used to determine the accuracy of the hypotheses generated by Chen et al.

most common words within a lexicon and sorted them based on their context to determine what their part of speech was. They applied their method not just to English, but also to the collected works of Cicero in Latin. I plan to attempt to implement a basic version of what they did in order to identify important characteristics of the words, such as case, tense, person, etc.

One method for translation used by Bowden covered the usage of possible word tags to eliminate possibilities of what role the word could fill in the sentence. This method allowed for faster translation and deduction of word order. I am implementing a similar method on a smaller scale grammar-wise. By combining this method with statistical translation strategies, I hope to be able to improve on Bowden's original research.

Other methods for translation have been tested to find a replacement for use of n-grams, particularly bigrams. One experiment by Pla et al. attempted to combine several methods for machine translation to create one that would be overall more efficient than bigrams, but their experiment showed that bigrams were still slightly more efficient than their own methods.

Alum et al. did an experiment with another language in which their system for statistical analysis was analyzing part of speech pairings. By doing this, they could eliminate some obviously bad pairings, such as putting the direct object after the subject. I will implement a similar method by comparing the parts of speech indicated by the Latin tenses, and then arranging the sentence based on this knowledge. I hope to improve on their original findings, as their attempt did not end up with very good results, largely due to the problem of the corpus that they used.

Another method for improving the base translations in conjunction with n-grams is to use Machine Assisted Translation, as used by Barrachina et al. By checking the first part of the translation and correcting for errors using user input, the computer is able to generate a better possible translation by correcting for any errors it may have had in the first part. While incorporating user input does result in a somewhat human-dependent program, the program is still much more efficient than a human simply doing the translation by hand. It also allows for more accurate translations generated by the n-gram statistical method.

Secunda legio castra in Gallia habet, sed in Britanniam cum imperatore festinabit.

STAGE 1 The output from the first stage is as follows:

Second[ADJ-NomSingFem,VocSingFem,AblSingFem,NomPlurNeut,VocPlurNeut,AccPlurNeut] **legions**[NOUNFEm-NomSing,VocSing] **camps**[NOUNNeut-NomPlur,VocPlur,AccPlur] **in**[PREP-+Abl]-OR-**into**[PREP-+Acc] **Gaul**[NOUNFem-NomSing,VocSing,AblSing] **he/she/it_have**[VERB-3rdSingPresIndAct] , **but**[CONJ] **in**[PREP-+Abl]-OR-**into**[PREP-+Acc] **Britain**[NOUNFem-AccSing] **with**[PREP-+Abl] **generals**[NOUNMasc-AblSing] **he/she/it_will_hurry**[VERB-3rdSingFutIndAct] .

Figure 3: An example of tagging for possible characteristics of words from an experiment by Bowden.

3 Design and Procedures

3.1 The Dictionary

My program used two dictionary data structures in order to translate the Latin sentences both accurately and quickly. The first dictionary was a regular Latin dictionary, which contains the important characteristics of the words as well as several possible meanings for each word. For this program I used the primary definition for each word. The second dictionary was an index of every word form translation that my program translated. By doing this, as the program translated more and more, it was be able to run faster and faster by simply looking up the indexed form of the word.

3.2 Word Tagging

Once the dictionaries were built and read, the program began its work on the input sentence. It read through each of the words and determined whether the word is a noun or a verb and then performs the tagging and translation methods for that type of word. The program identified the word's primary characteristics by either locating its genitive form, for nouns, or its infinitive form, for verbs, in the dictionary. Once it did so, it could eliminate all other possible declensions or conjugations and only focus on the specific one. It then iterated over the set of possible endings that that particular declension or conjugation could have. If the word ended in one of those endings, then my program affixed a tag to the word, identifying what that form's Case/Number/Gender or Person/Number/Tense was. The program repeated, identifying all the possible tags for that word.

3.3 Translation

Once the program finished tagging the word, it moves on to the translation stage. The program traversed over the list of tags for each word and generated a basic translation based on the information in the tag.

```
>>> ================================ RESTART ================================
>>>
What sentence would you like to translate?amabat
Time to make the dictionary is: 3.00600004196
Is this translation correct: 1IS3:He was loveing? no
Please enter correct translation now: he was loving
Time to tag sentence is: 4.32699990273
{'amabat': [['1IS3', 'he was loving']]}

Translation time is: 0.0199999809265
Total time taken is: 7.40799999237
>>> ================================ RESTART ================================
>>>
What sentence would you like to translate?videbit
Time to make the dictionary is: 2.76999998093
Is this translation correct: 2FS3:He will see? yes
Time to tag sentence is: 3.00900006294
{'videbit': [['2FS3', 'He will see']]}

Translation time is: 0.0209999084473
Total time taken is: 5.85700011253
```

Figure 4: A screenshot of the code as it prints out now

In addition, once the translation had been generated, the program asked the user if the output translation was correct. This way the program could correct for irregular forms of words, such as 'have' and 'has', without having to deal with the exceptions manually Once the forms have been translated, they were recorded into a secondary dictionary for easy access later. This saved the program

time the more it translates as it was able to simply recall the translated forms instead of having to translate them and correct them again.

4 Results and Discussion

The program was successfully able to tag and translate all noun and present stem verb forms without incident. In addition, it could read from the secondary dictionary in order to recall words that had already been translated before. Although I was not able to complete the translation part of the program, the work that I have done on this project leaves it available to easily implement the n-gram translation. In the future, additional work could be done to implement the machine-assisted translation along with the n-gram translation. More grammatical forms could also be implemented to allow for greater translational capability.

References

[1] Boxing Chen, Min Zhang, and AI TI AW., "Two-Stage Hypotheses Generation for Spoken Language Translation", *ACM Trans. Asian Lang. Inform. Proccess. 8, 1*, Article 4, 22 pages March 2009

[2] J. McMahon and F.J. Smith "Structural Tags, Annealing and Automatic Word Classification", *Sruct. Tags, Word Class.*, Queen's University of Belfast, May 1994

[3] Paul R. Bowden, "Latin to English Machine Translation - A Direct Approach", *The Machine Translation Review* Issue 12, December 2001

[4] Sergio Barrachina, Francisco Casacuberta, Elsa Cubel, Antonio Lagarda, Jesus Tomas, Juan-Migel Vilar, Oliver Bender, Jorge Civera, Shahram Khadivi, Hermann Ney, Enrique Vidal, "Statistical Approaches to Computer-Assisted Translation", *ACM*, 26 pages September 2007

[5] Ferran Pla, Antonio Molina, Natividad Prieto, "Tagging and Chunking with Bigrams", *Universitat Politbcnica de Val5ncia*, Departament de Sistemes Informatics i Computaci6