Tagging and Statistically Translating Latin Sentences

Andrew Runge

April 6, 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.

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.

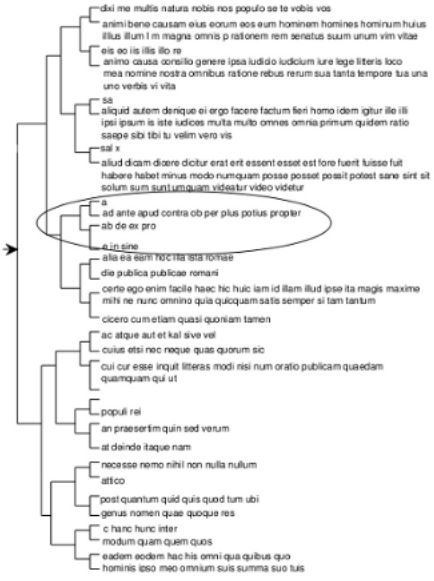

Figure 1: Tree of words sorted by sentence role from the assorted works of Cicero generated by the methods of McMahon and Smith.

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. However, 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. 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.

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.

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

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

McMahon and Smith also demonstrated use of n-grams in word tagging for part of speech purpose. They generated trees of the 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 in part due to the problem of the corpus that they used.

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 uses two dictionary data structures in order to translate the Latin sentences both accurately and quickly. The first dictionary is a regular Latin dictionary, which contains the important characteristics of the words as well as several possible meanings for each word. For the sake of this program, I will be using the primary definition for each word. The second dictionary is an index of every word form translation that my program will have translated. By doing this, as the program translates more and more, it will be able to run faster and faster by simply looking up the indexed form of the word. This will be most effective in translating large bodies of text, as it will greatly increase the size of this second dictionary and improve the program's proficiency.

3.2 Word Tagging

Once the dictionaries have been built and read, the program begins its work on the input sentence. It reads through each of the words and determines whether the word is a noun or a verb and then performs the tag-

ging and translation methods for that type of word. The program identifies the word's primary characteristics by either locating its genitive form, for nouns, or its infinitive form, for verbs, in the dictionary. Once it has done so, it can eliminate all other possible declensions or conjugations and only focus on the specific one. It then iterates over the set of possible endings that that particular declension or conjugation can have. If the word ends in one of those endings, then my program affixes a tag to the word, identifying what that form's Case/Number/Gender or Person/Number/Tense are. After identifying all the possible tags, the word goes on to the translation method.

3.3 Translation

After finishing the tagging, the word moves on to the translation method. The translation method simply takes each of the word's tags and generates the translation for that form of the word. It stores all of these forms into the secondary dictionary once it has created them, so that it can later simply recall them should that form of the word come up again. The process repeats for each word in the sentence until every word has been tagged and translated in all its possible forms.

4 Expected Results and Discussion

I expect that my program will be able to properly translate Latin sentences using the methods detailed in my introuction and design procedures. I will test the program on several sections of Latin, ranging from very basic sentences to original Latin prose. Right now, my program can currently fully translate all nouns, and tag all verbs. Once I finish the translation for verbs, the next stage will be to work on the ordering of the words in the sentence. After that, I will concern myself with adding in any additional grammar that I can in order to improve the range of material that my program can translate. Further research in this area can be done to improve the use of n-grams and their efficiency. In addition, further research can be done in improving statistical methods used to generate the hypotheses for translation.

```
>>> ================================ RESTART ================================
>>>
What sentence would you like to translate?puellis puerum rei
Time to make the dictionary is: 1.71600008011
Time to tag sentence is: 0.0
{'puerum': [['2SA', 'boy'], ['2SV', 'boy']], 'puellis': [['1PD', 'to the girls']
, ['1PB', 'the girls']], 'rei': [['5SG', 'of the thing'], ['5SD', 'to the thing'
]]}

Translation time is: 0.0
Total time taken is: 1.77800011635
```

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

References

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

[2] J. McMahon and F.J. Smith "Struc-

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