Machine Learning, Language Rules, and Statistical Translation

Andrew Runge

January 19, 2010

Abstract

Development of language translators, spoken or written, has most often used either rule-based or statistical strategies. In addition, machine learning is becoming one of the most efficient and effective methods for interpreting and deciphering text. Through the use of machine learning, the less common rule-based strategies may be implemented to greater effect. This project aims to use machine learning strategies to combine these two strategies to create an effective and efficient Latin translator. The project will be tested on several samples of Latin, including original Latin prose. The results will be studied for correct grammar, as well as compared to human translation of the same lines. The program will be done using python and the IDLE interface.

Keywords: Machine Learning, Statistical Translation, N-Gram

1 Introduction

The field of machine translation has been growing significantly over the past few years as a way to advance artificial intelligence and to make computers more capable of operating on their own in the real world. One such example of machine translation is through language translation software. Language translators have often been developed using two different strategies. Rule-based strategies are use to translate words properly so that their purpose in the sentence can be accurately defined. In addition, rule-based strategies have been used to try to properly put words in an intelligent order so that the resulting sentence makes sense. However, rule-based strategies are often not proficient in assigning word order. The use of n-grams to determine likely sentence structures results in a more logical word order and is more efficient at assigning that word order than rule-based strategies are.

N-grams are sets of words in a sentence, n words long, which can be used to identify word context, and from this its role in the

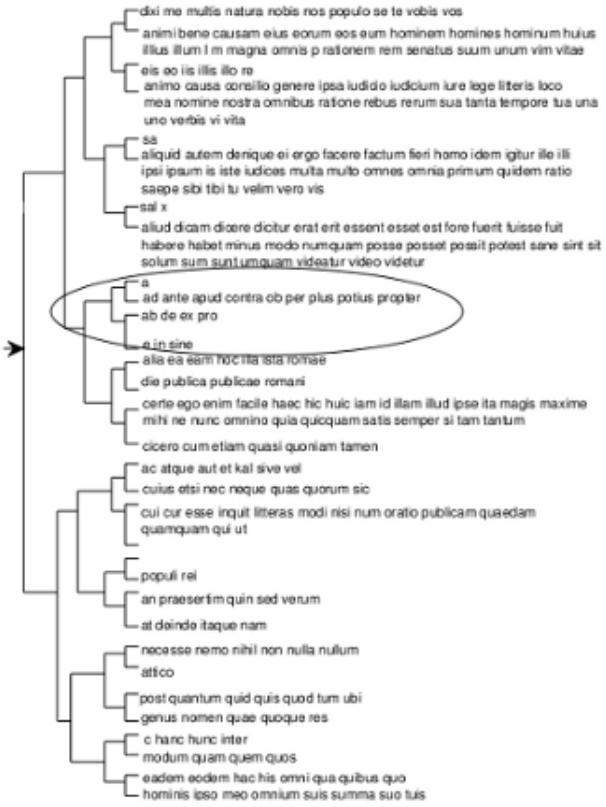

Figure 1: Tree of words sorted by sentence role from the assorted works of Cicero generated by the methods of McMahon and Smith.

sentence. In addition, n-grams are often used in the process of tagging words for things such as part of speech. Using n-grams and machine learning, it is possible to quickly and easily translate a sentence, but there still remains the problem of getting the word order in the sentence correct. This is where the second method, statistical analysis, comes into play. Statistical methods also make use of n-grams, but to a different effect. They use the n-gram and find the most common contexts of that word in various other situations to then generate partial n-grams of the given words in order to improve word order and subsequently sentence interpretation.

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.

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.

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.

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

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

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

The first thing that my program must be able to do is correctly identify all the important characteristics about each word. For nouns, this would mean case, number and gender. For verbs, this would mean person, number, tense. For the purpose of this experiment, I am disregarding voice and mood unless time permits added functionality to translate them. In order to determine these characteristics of the word I will be using a word tagging algorithm in order to determine all possible combinations of characteristics that a word can have. The algorithm will narrow down the possibilities through analyzation of different aspects of the sentence, such as making sure the subject and verb agree in number, making sure adjectives agree with their subjects, etc. By doing this, there will remain a very limited number of possible tags which remain. My code will then look at a Latin corpus to determine how it can further reduce the possible tags for each word. This will include looking at what cases verbs take and what how frequently a word shows up in a given case. Once I have reduced all the possibilities, the final stage will be to use an n-gram reader to analyze an English corpus and use the information to generate hypotheses based on part-of-speech proximity analyzation. Basically, the program will read an English corpus, the Brown corpus, and determine probabilities that two parts of speech are next to each other. This way, my program will be able to eliminate completely non-sensical translations from the possible hypotheses that it generates.

4 Expected Results and Discussion

I expect that my program will be able to properly translate Latin sentences using the methods detailed in my introduction and design procedures. I will test the program on several sections of Latin, ranging from very basic sentences whose words are already in the correct English order, all the way up to original Latin prose. As it stands, my program currently can translate individual words, but cannot yet translate full sentences in an intelligent manner. In addition, my program can also correctly identify all possible noun characteristics. The next step will be to apply that to verbs and then I will begin actual translation stage, adding more types of grammar and parts of speech once I have that section working. 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.

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