Latin Word Segmentation Model Comparison

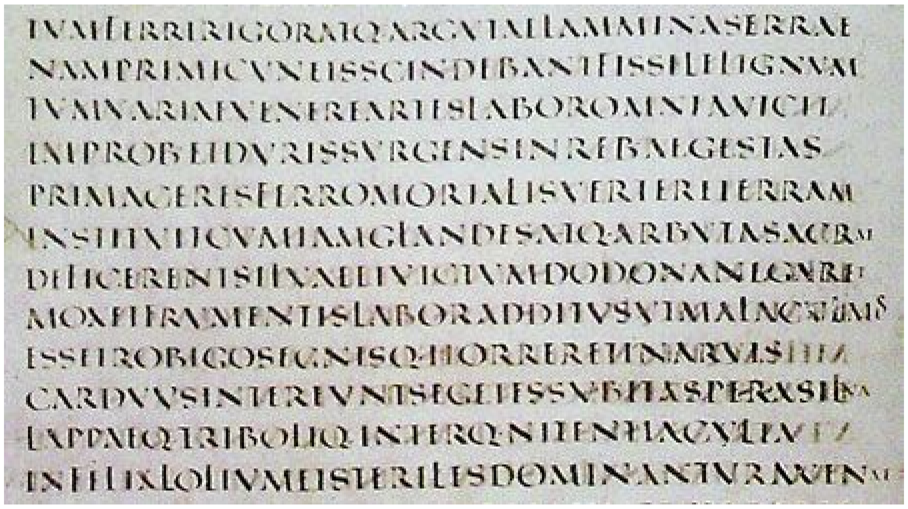
Eric Bailey
December 22, 2015
eric.bailey@tufts.edu

Abstract

For this paper, I will give my results of a comparison of various methods of programmatically deciding where to place word segmenters given unsegmented Latin text. I compared 3 models; a supervised, intuitive N-gram metric, a supervised Maximum Word Matching metric designed for Chinese word segmentation (Wong and Chan, 1996), and an unsupervised metric called the TANGO algorithm designed for Japanese word segmentation (Ando and Lee, 2003). Not too surprisingly, the supervised methods performed better than the unsupervised methods, but the TANGO algorithm still performed rather well. Compared to a unigram-based baseline measure, the 3 algorithms in question performed much better.

Introduction

In Ancient Rome, Latin text was often written with word separators (an *interpunct* in the middle of the line) rather than spaces. However, around 200 CE, writers started skipping the *interpuncts* and just wrote without segmenting their words - this style of writing is called *Scriptio Continua*. While most of the texts we currently know of are already parsed for spaces, if we find a large body of new Latin text, we could speed up understanding/parsing of this text by applying a programmatic approach rather than having translators trying to semantically understand the Latin. Still, it is still interesting/useful to see what NLP methods work well under which circumstances.


TVMFERRIRIGORATQ·ARGVTAELAMMINASERRAE
NAMPRIMICVNEISSCINDEBANTFISSILELIGNVM
TVMVARIAEVENEREARTESLABOROMNIAVICIT
IMPROBVSETDVRISVRGENSINREBVSEGESTAS
PRIMACERESFERROMORTALISVERTERETERRAM
INSTITVITCVMIAMGLANDESATQ·ARBVTASACRAE
DEFICERENTSILVAEETVICTVMDODONANEGARET
MOXETFRVMENTISLABORADDITVSVTMALACVLMOS
ESSETROBIGOSEGNISQ·HORRERETINARVIS
CARDVVSINTEREVNTSEGETESSVBITASPERASILVA
LAPPAEQ·TRIBOLIQ·INTERQ·NITENTIACVLTA
INFELIXLOLIVMETSTERILESDOMINANTVRAVENAE

There is relatively little work being done on Latin word segmentation compared to other languages (such as Chinese and Japanese). Also, evaluation metrics for other languages may not be comparable; for example, Chinese words only contain up to four characters, whereas Latin words can be any number of characters. For this reason, the maximum matching segmentation method for Latin words may provide words that are too long than the correct parsing, when it may be more accurate for Chinese. Also, Chinese and Japanese have thousands of characters that could comprise potential words, whereas Latin just has 23 characters (all of our upper case characters, without W, J, or U), so we run into different sparsity problems than other modern segmentation applications, which normally use non-Latin characters.

To compare the models, I used all the same data, removing the spaces in the already-segmented data for the unsupervised model. I used a manually-gathered corpus of 11,534 sentences, and used the first 10,000 sentences for the training set and the 1,534 for the test set. The Baseline performed with an F-measure of around 0.28, TANGO performed with F-measure of 0.56, the N-Gram method performed with an F-measure of 0.63, and the maximum matching performed with an F-measure of 0.65, based on my own evaluation metric (explained below).

Methods

Baseline

The baseline approach was quite simple. After training on the supervised corpus of sentences/words, given a new unsegmented sentence and index *k*, if *sent[k-1]* was the end of a word we've already seen and *sent[k]* is a valid beginning of a word we've already seen, then we place a space before *sent[k]* in our parsed sentence. Because of the numerous possible word boundaries, this technique tended to place too many spaces, resulting in a high recall but low precision. This baseline method was taken from Dylan Rhodes, 2013. For this baseline, the overall F-Measure for the test set was 0.28. There are many words that start and end with almost all combinations of characters, so this baseline method produced sentences with far too many spaces, resulting in an average recall of 0.16, but precision of 1.0.

TANGO

I implemented the unsupervised TANGO algorithm described here by Ando and Lee in 2003. Although unsupervised, TANGO performed much better than the baseline, reporting an F-measure of 0.56. The original TANGO paper claimed to produce results better than that of many modern supervised segmentation methods. However, this result is domain-specific (because they were designing an algorithm for specifically Japanese Kanji sequences). This is also reflected in my results, as the supervised methods both gave better results than TANGO. TANGO resulted in a recall of 0.68 and precision of 0.48, because it tends to produce sentences with too many spaces rather than too few.

N-Gram

I also came up with my own N-gram method of segmentation. In it, we essentially use TANGO's "vote" idea, placing word boundaries at local maxima or those above a certain threshold, but rather than using an unsupervised vote method, we use a rather intuitive supervised approach. If the (N/2)-grams to the left and the right are more likely to occur with a space between them than with a space, then we place a space there. For example, if "ABCD 1234" is more likely than "ABCD1234", then we would place a space in our new sentence at location 4, before the '1'. Because of the supervised nature of this metric, it performed better than TANGO, reporting an F-Measure of 0.63. The precision and recall were rather similar, at 0.624 and 0.66, respectively.

Maximum Matching

As a final metric, I implemented the supervised Maximum Matching method described here by Wong and Chan in 1996. Its idea is also rather simple: we keep a current valid sub-word initialized at the empty string, and if adding the next character in the sentence to the currently kept word causes it to become invalid, then we add a space there and restart the "current word" at the empty string. For example, parsing "HANDSTAND" would produce "HANDS TAND", but not "HAND STAND", because "HANDS" is part of a valid English word, but "HANDST" is not. For efficiency, I stored the word dictionary in a trie, which resulted in very fast word lookup. This metric performed slightly better than my N-Gram method, reporting an F-Measure of 0.646. Because this method tends to create words that are too long, the recall was rather high at 0.72, with the precision at 0.59.

Experimental Setup

Luckily, to get supervised data for this project, I just needed to get already parsed Latin sentences, not the English translations of those sentences. To do this, I downloaded all of the texts from a project called Libellus, spot checked for accuracy (*Libellus* means "little book" in Latin, and as a fun fact, this is probably a reference to Catullus 1).

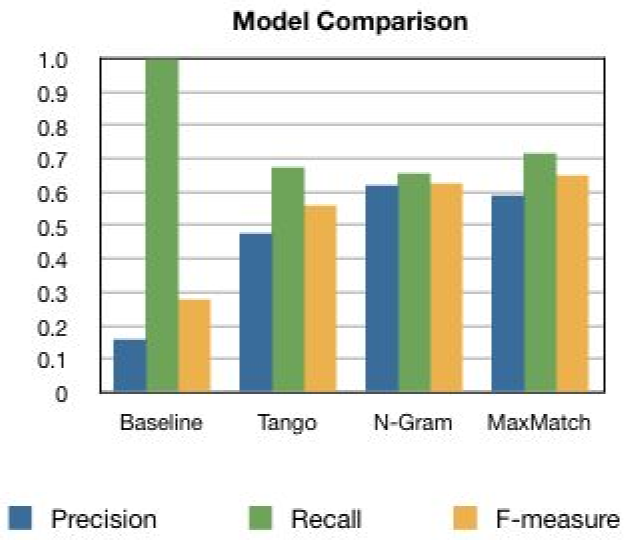
Text preprocessing was required for this project - I parsed all the paragraphs into individual sentences, and converted them into a form that was most like it was originally written (made all characters uppercase (Latin didn't have lowercase letters), replaced "J" with "I", and replaced "U" with "V" (Latin used these characters synonymously)) in order to get more accurate/original data and to decrease sparsity.

Both TANGO and my own N-gram method only require N-grams, so I used NLTK's built-in ngram and everygrams functions to get all the required N-gram data, using 2 passes (one for the supervised data and one for the unsupervised data). BeautifulSoup was used to parse the raw HTML data from Libellus. No other tools or packages were used.

My evaluation metric was simple: I sequentially walked along the non-space characters of both the correct and parsed sentence. If a space was supposed to be there that wasn't, a false negative was accounted for; if a space wasn't supposed to be inserted but was, a false positive was accounted for; if a space was correct inserted, a true positive was accounted for. The F-measure, precision, and recall were based on these metrics of false/true positives/negatives. This may be a different way of accounting for true/false positives/negatives than other papers, so direct comparisons to these (regarding F-Measure) cannot be made.

Results

As mentioned previously, the supervised methods performed much better than both the baseline and the unsupervised TANGO algorithm. Here is a visual representation of the results:



Based on the F-measure performance of TANGO, we can determine that the TANGO algorithm is domain-specific to Japanese Kanji sequences and not as applicable to other domains, as TANGO performed better than supervised methods for Japanese Kanji datasets, but not necessarily for Latin. Latin is different than Japanese in that the number of possible characters that can comprise words is far fewer (23 as compared to thousands of possible characters). Also, for Latin specifically, for an optimal segmentation algorithm, domain-specific knowledge (that takes into account, for example, the few number of vocabulary characters compared to Asian languages) may need to be applied rather than language-agnostic approaches to achieve the best accuracy. The TANGO algorithm may be similar for segmenting other languages which may have a large vocabulary of possible characters (such as information extraction, even), but because of the differences in the languages, better approaches to segmentation can be applied to Latin than the TANGO algorithm, which was designed with Japanese in mind.

Future Work

There are many other ways in which NLP approaches could be applied to understanding original Latin text. Consider the original Latin phrase written in *Scriptio Continua*: "C.IVLIVSCAESAR". In parsed modern Latin text, the phrase would be written as "C. Julius Caesar". Notice the ambiguity - one of the I's was correctly written as J, but not the other. The same happens with "V" versus "U". Similarly, only some of the letters were put into lowercase.

As previously mentioned, Latin only had 23 characters; all of our upper case characters, but "I" and "J" were both written as "I", both "V" and "U" were written as "V", and "W" was not used. We could use modern spelling correction methods to decipher which instance of V/U or I/J should be used, and we could also apply spelling correction and n-grams to the upper/lowercase problem.

Also, translation of original Latin text into English could be applied - almost certainly, correct parsing of the Latin into modern readable Latin text would improve automated translation efforts. Thus, this would prove very useful for automated understanding of original Latin artifacts and ensuring that the parses/translations we trust today are correct.

References

Rie Kubota Ando and Lillian Lee. *Mostly-Unsupervised Statistical Segmentation of Japanese Kanji Sequences*. Cambridge University Press, 2003. http://www.research.ibm.com/talent/documents/ando-lee-nle03.pdf

Dylan Rhodes. *Conditional Random Field Latin Word Segmenter*. Stanford University, 2013. http://nlp.stanford.edu/courses/cs224n/2013/reports/dylanr.pdf

Pak-kwong Wong and Chorkin Chan. *Chinese Word Segmentation based on Maximum Matching and Word Binding Force*. The University of Hong Kong, 1996. http://www.aclweb.org/anthology/C96-1035