Conditional Random Field Latin Word Segmenter

Dylan Rhodes (dylanr)

December 8, 2013

Abstract

This project describes the implementation and results of two statistical learning models for Latin word segmentation. Segmentation of Latin words is an interesting topic, because although Romans initially recorded their writing with interpuncts between words, that habit died out for several centuries of the first millenium AD when a style known as *scriptio continua* became more prevalent. First, motivation for the work and selection of the dataset are explained, followed by a brief explanation of the two segmentation models. Next, results of the baseline dictionary model are compared with those of the conditional random field model as feature engineering is performed. Finally, the overall results and remaining error types are discussed and some suggestions are offered for further model improvement.

1 Roman History Background

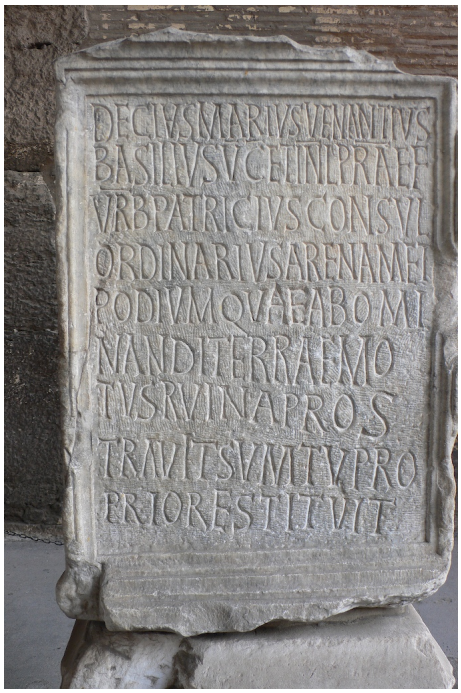

Figure 1: *Scriptio continua* in the Roman Collosseum circa 484 AD.

Feature based word segmentation is a field of study in natural language programming which has seen significant research in recent years. Most of that effort has been directed towards modern, written languages such as Chinese, Arabic, Thai, and Vietnamese, in which word dividers are traditionally omitted. However, omission or absence of word dividers is an issue for several historic languages as well, including Roman Latin.

In the ancient Mediterranean world, many languages recorded the written word with the Phoenecian alphabet, which did not include glyphs for vowel sounds. Word dividers, commonly vertical lines or interpuncts, were necessary to keep the text intelligible. However, with the introduction of vowel characters in the Greek alphabet, word dividers became less important and were soon abandoned by the Greeks. Interpuncts remained in use by the Romans until the end of the Classical period c. 200 AD, when the Greek style of *scriptio continua* became more fashionable. Most documents recorded between this time and the 7th century AD, when whitespace delineation began to emerge, employ the *scriptio continua* style. Thus, automatic word segmentation of Classical and Post-Classical Latin text is a potentially useful tool for analysis of surviving work from this era.

2 Preprocessing and Selection of the Dataset

Author	Works included
Albius Tibullus	*Aliorumque Carminum*
Publius Vergilius Maro	*Aeneid, Bucolics, Eclogues*
Gaius Valerius Catullus	*All Surviving Works*
Sulpicia	*Epistulae*
Claudius Claudianus	*Panegyricus Dictus et Sextus*
Sextus Propertius	*Elegiae*
Gaius Valerius Flaccus	*Argonautica*
Publius Ovidius Nason	*Metamorphoses Liber I-X, Amores, Ars Amatoria*
Phaedrus	*Fabularum Aesopiarum*
Quintus Horatius Flaccus	*Carmina*
Publius Papinius Statius	*Achilleid*

Table 1: List of works included in the dataset.

The dataset was chosen with the c. 200 to 700 AD timeframe in mind. Care was taken to select authors of the early centuries AD whose work would have likely been transcribed in *scriptio continua*. In particular, the work of noted poets was included in the dataset, both because it, unlike Latin prose, is delineated into lines, and because it generally would have been written on scrolls for oration in a *scriptio continua* format. Ultimately, assorted works from eleven noted Roman poets comprising over forty-six thousand lines were included in the dataset.

Preprocessing of these texts was necessary before any analysis could be performed. Since the classical period, scholars have long since segmented, capitalized, and punctuated these texts for the consumption of modern audiences, even in written Latin versions. A simple script was written to parse these additions from the dataset and label each capitalized Roman character with a '1' for end of word and '0' for extension of word. In this way, the word segmentation features of the modern texts were preserved as a gold standard for model training, but all other introduced, modern features were excluded. The final result resembles the declamation inscribed in Figure 1: several lines as sequences of capital letters.

3 Explanation of Statistical Models

Two types of statistical model were examined and their results compared over the course of this project. On the advice of Professor Manning, a naive baseline model was implemented as a simple dictionary of words from the training data. In addition, an iterated conditional random field model with different feature sets was evaluated.

3.1 Baseline Dictionary Segmentation Model

The baseline model was fairly simple, yet proved useful enough to establish some results. In the training phase, the training dataset is iterated through and each unique word stored in a HashSet dictionary. Then, during the test phase, the algorithm greedily checks whether the substring preceding the current character up to the last marked end of word character or beginning of the string is contained in the dictionary. To avoid total failure after a single missed ending on a given line, after eight characters marked word extension the algorithm decides to end the word. The max length eight was chosen through cross validation, but only the final results of the baseline classifier are ultimately important for comparison to the CRF model.

3.2 Conditional Random Field Segmentation Model

The Conditional Random Field (CRF) segmentation model made use of a C++ implementation called crfsuite and authored by Professor Naoaki Okazaki of Tohoku University. This library imports feature vectors as tab separated lists of string features with the label value as the first string. Thus, another tool was written to automatically generate feature vector files for quick feature engineering with crfsuite.

CRFs are a class of statistical learning model which has been applied to the task of word segmentation with success in the past. In particular, the Stanford Chinese and Arabic Word Segmenters both employ CRF classifiers over various feature sets for segmentation. CRFs are undirected, probabilistic graphical models which learn weights through numerical optimization based on a combination of features of the current datum and preceding, already-classified data. The model employed in this project was a first order Markov CRF, so it considered only transitional features from the directly preceding character. For iterative numerical optimization of the parameter weights, the quasi-Newton, Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm was applied, which is a modified gradient descent algorithm.

BFGS Algorithm

From initial guess x_0 and initial matrix $B_0 = I * x_0$ repeat steps as $|\nabla f(x_k)|$, the norm of the gradient of the objective function to be minimized, converges.

1. Obtain a direction p_k by solving $B_k p_k = -\nabla f(x_k)$
2. Perform line search (More-Thuente algorithm in this case) to select a reasonable update coefficient α_k and update $x_{k+1} = x_k + \alpha_k p_k$
3. Set $s_k = \alpha_k p_k$
4. Set $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
5. Then, $B_{k+1} = B_k + \frac{y_k y_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k} - \frac{B_k s_k s_k^{\mathrm{T}} B_k}{s_k^{\mathrm{T}} B_k s_k}$.

---

4 Discussion of Test Methods

Testing was performed in a rigorous manner. First, the dataset was randomly shuffled and partitioned into a training set comprised of four fifths of the overall dataset and a held out test set of the remaining fifth. During the feature engineering process, a smaller evaluation set was drawn from a randomly sampled third of the training data. Features which failed to increase the average precision and recall after three fold cross validation over this evaluation set were excluded from the final model. Three fold cross validation was employed for this process due to the lengthy training time of cross validation methods with a greater number of folds. The first iteration of the CRF based model with a fairly naive feature set outperformed the baseline in these cross validation results. Further, successive addition and removal of features managed to improve the results of the CRF segmentation model significantly. After the optimal feature set had been determined through cross validation on the evaluation set, the final result was derived from a model which was trained on the full training dataset and evaluated on the test set, which had not been touched since partitioning. Keeping the test data completely held out of the feature engineering process ensures that the final results were not a result of overfitting to the test data, but rather reflect the strength of the linguistic assumptions implied by the features included in the final model.

5 Feature Engineering

Feature Set Added	Precision	Recall	F1 score	Included
Unigrams [-2,2]	.812	.768	.789	Yes
Character Indices	.817	.784	.800	Yes
Previous Bigrams	.847	.826	.836	Yes
Next Bigrams	.893	.873	.883	Yes
Trigrams	.914	.899	.906	Yes
isPrefix	.914	.899	.906	No
isVerbEnding	.914	.899	.906	No
isNounEnding	.914	.899	.906	No
Expand n-grams to [-3,3]	.948	.940	.944	Yes
4-grams	.957	.947	.952	Yes
5-grams	.959	.949	.954	No
6-grams	.959	.948	.953	No

Table 2: Three fold cross validated results of iterated feature engineering.

There were roughly two classses of feature explored over the course of feature engineering: n-grams and grammatical tests. The motivation for inclusion of these two classes came directly from previous work in word segmentation through the Stanford Chinese and Arabic Word Segmenters. From the code of

the Chinese segmenter, one can see that the feature set is primarily comprised of n-grams in various permutations, as well as part of speech, named entity type, punctuation, numerical, and foreign language features. However, since Latin characters do not comprise whole words, the concept of part of speech or named entity type does not hold for an individual character. Further, the significantly simplified Latin dataset excluded Arabic numerals, punctuation, and other languages. Thus, the only cross-applicable class of features from Chinese turned out to be n-grams.

The Arabic segmenter yielded a different set of features for analysis. Without delving into too much detail, there are a class of characters in Arabic called bound morphemes which are generally indicative of a word boundary. This concept is loosely extendable to the Latin language, although there are no syllables which definitively mark the end of a word. However, there is a finite set of verb conjugation endings and noun endings based on part of speech and declension. The use of word endings to denote meaning is crucial in Latin, since there are no grammatical rules governing word order. A sentence can be randomly permuted and retain the same meaning, since each word is marked with its usage by its ending. This is particularly crucial in poetry, the subject of this project's analysis, since Roman poets often permuted their words as literary devices or to maintain a certain meter. This train of thought lead to the isVerbEnding and isNounEnding features above, which looked for matches backward from the current character among a list of common verb and noun endings respectively. Further, many Latin words begin with a slightly less well defined set of prefixes generally derived from prepositions. This tendency can be observed in many English words with Latin origins ie. exit, expel, extract, and except from the Latin prefix ex-, meaning out of. The prevalence of preposition-derived prefixes is even greater in Latin itself, which motivated the isPrefix feature above. The remaining features in the Arabic segmenter are variations on the n-gram theme, which would prove to be key to the final model.

From the table of cross validation results above, it is clear that the n-gram features' performance dominates that of the more linguistically based features. Indeed, variations of the three linguistic rules, which were expected to be quite predictive, proved to have no significance whatsoever in the model's accuracy. Although this result disagrees with the initial hypothesis of their usefulness, it does make some intuitive sense. Upon analysis of the parameter weights, one can see that many of the endings which compose the verb and noun ending lists take on much higher weights than average. In fact, although the mean magnitude of all parameter weights is only 0.122, the mean magnitude of weights of features with n-grams contained in the verb and noun endings list are 0.775 and 0.849, respectively. This suggests that the linguistic assumptions which had been attempted to be expressed by the isVerbEnding and isNounEnding features were valid, but that the n-gram features proved much more expressive of the underlying assumptions.

Finally, both the number of preceding characters and number of remaining characters in the sequence were included as features, exhibiting a slight improvement in accuracy. However, it is extremely likely that this improvement is due

to the fact that all final characters must be the end of a word, since words cannot extend across two lines in Latin poetry. Indeed, the single highest weighted feature proved to be the indicator that a character was last in its sequence. This can be said to take advantage of a linguistic assumption, but it is an obvious one at best.

6 Final Results

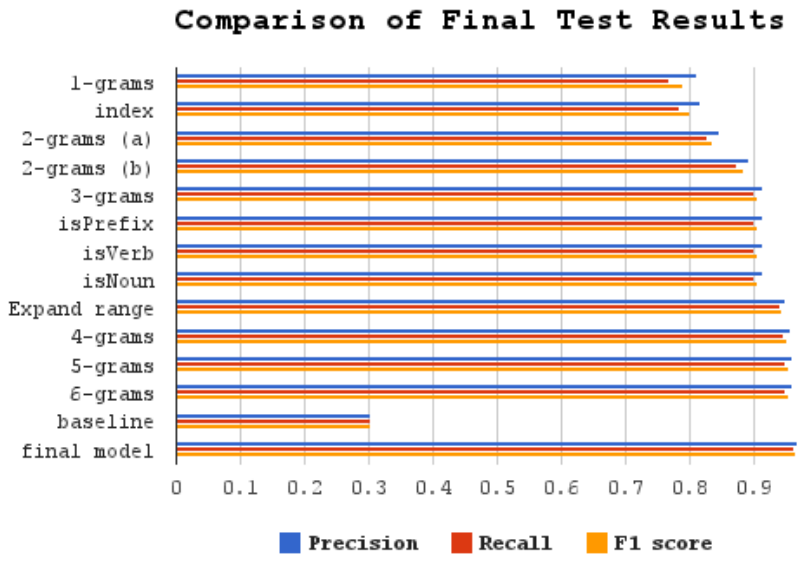

Figure 2: Visual comparison of cross validation, baseline, and final test results.

Model	Precision	Recall	F1 Score	Accuracy
Baseline Dictionary Model	.302	.303	.302	75.18%
Final CRF Model	.968	.962	.965	97.95%

Table 3: Final Baseline and CRF results over entire dataset.

The final results of the project were better than expected. There was a slight gain in both precision and recall in the final test run of the CRF model. This is almost certainly due to the inclusion of the entire training dataset in the training phase, rather than the third which had been sampled for cross validation and feature engineering. One truism of statistics is that expanding the number of samples in the dataset almost always improves results. The final F1 score was on par with that of the Stanford Chinese word segmenter, and the character classification accuracy approached that of the Arabic segmentation model at 97.95% to 98.6%. It would likely be possible to improve upon these metrics

through further expansion of the dataset and additional feature engineering, but the performance of the model as-is was extremely satisfactory.

There are some remaining error types which might avail themselves to further feature engineering. In particular, the inherent ambiguity of common, concatenated words in the Latin language poses a difficult challenge for the word segmenter. As an example, the common Latin word *quoque* is comprised of two separate, also common words, *quo* and *que*. The Latin segmenter has difficulty distinguishing this word from its components and often misclassifies them. The situation is somewhat ameliorated by the addition of longer n-grams, but deeper linguistic features could probably approximate the problem better. Often, compound words consist of two words which are grammatically unlikely to follow each other, such as a preposition followed by a verb eg. *permitto* = *per* + *mitto*. A feature which could capture this tendency in some way would be non-trivial to implement, since it would likely have to employ a hand written dictionary of parts of speech to compare with the possible segmentations of a given phrase. However, it has the potential to partially solve the ambiguity posed by compound words.

7 Conclusion

Overall, this project illustrates the strength of both low level features and well selected statistical models. CRFs and sequence models in general are extremely well suited to the task of statistical word segmentation. Admittedly, the task of Latin word segmentation is in and of itself slightly facile, since all surviving Roman texts have long since been hand segmented by classics scholars as a first step in translation. The final CRF model trained in this project could offer some utility to a Latin scholar in the field, however. Anecdotally, if applied to a transcription of the text shown in Figure 1 at the Roman Collosseum, the model achieves 95.24% classification accuracy with an F1 score of 0.914, segmenting all but the last line perfectly. Instantaneous segmentation of Latin words could greatly speed one's translation of Roman words, since a significant amount of time reading Roman inscriptions is often spent deciphering word boundaries. The power of modern computing paired with valid statistical reasoning is astounding.

References

[1] Chang, Pi-Chuan, Michel Galley, and Christopher D. Manning. "Optimizing Chinese word segmentation for machine translation performance." *Proceedings of the Third Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2008.

[2] Green, Spence, and John DeNero. "A class-based agreement model for generating accurately inflected translations." *Proceedings of the 50th Annual*

Meeting of the Association for Computational Linguistics: Long Papers-Volume 1. Association for Computational Linguistics, 2012.

[3] Okazaki, Naoaki. "CRFsuite: a fast implementation of conditional random fields (CRFs)." *URL http://www.chokkan.org/software/crfsuite* (2007).

[4] The Latin Library. "*Classical Works*." Web. *http://www.thelatinlibrary.com*.