A Memory-Based Lemmatizer for Ancient Greek

Corien Bary
Radboud University Nijmegen
Department of Philosophy, Theology and Religious Studies
P.O. Box 9103
Nijmegen, The Netherlands 6500 HD
c.bary@ftr.ru.nl

Peter Berck
Radboud University Nijmegen
Centre for Language & Speech Technology
P.O. Box 9103
Nijmegen, The Netherlands 6500 HD
p.berck@let.ru.nl

Iris Hendrickx
Radboud University Nijmegen
Centre for Language & Speech Technology
P.O. Box 9103
Nijmegen, The Netherlands 6500 HD
i.hendrickx@let.ru.nl

ABSTRACT

In this paper we present the lemmatizer that we developed for Ancient Greek: GLEM. As far as we know, GLEM is the first publicly available lemmatizer for Ancient Greek that uses POS information to disambiguate and that also assigns output to unseen words, words that are not yet in the lexicon.

As the basis for the lemmatizer we used an existing memory-based learning tool, Frog, that was originally developed for Dutch and that we converted to work for Ancient Greek. As the results of Frog on Ancient Greek were rather modest, we used Frog to create a smarter lemmatizer, GLEM, that uses a lexicon look up in addition to the memory-based tool Frog. We evaluate and compare the performance of GLEM against the Frog lemmatizer and the already existing CLTK lemmatizer and observe that GLEM achieves the highest accuracy of 93% on an unseen test corpus sample. GLEM's look up component overcomes the difficulty of a relative small training set in combination with a morphologically rich language, while the memory-based learning component enables GLEM to handle unknown words.

CCS CONCEPTS

•Applied computing → Document preparation;

KEYWORDS

Ancient Greek, NLP, lemmatizing, POS tagging, memory-based learning

ACM Reference format:
Corien Bary, Peter Berck, and Iris Hendrickx. 2017. A Memory-Based Lemmatizer for Ancient Greek. In *Proceedings of DATeCH2017, Göttingen, Germany, June 01-02, 2017,* 5 pages.
DOI: http://dx.doi.org/10.1145/3078081.3078100

1 INTRODUCTION

Research into older stages of languages such as Ancient Greek is corpus-based by necessity. This type of research has been supported by the long-time availability of digitized versions of the relevant texts (Thesaurus Linguae Graecae [3], The Perseus Digital Library Project (Perseus for short) [5]), which are extensively used in the field of classical philology/linguistics. Recent years have seen a growing interest and endeavor to go a step further and apply more advanced corpus-based methodologies and standards known from computational linguistics to Ancient Greek (e.g. [9], [8] in the PROIEL project, [13] in Perseus, [7] in Perseus under PhiloLogic). Indeed such methodologies are bound to provide highly interesting and relevant new insights for linguists, literal scholars and historians alike. We contribute to achieving this potential by developing a lemmatizer for Ancient Greek: GLEM. This lemmatizer on the one hand helps scholars of the Greek language to uncover statistical patterns gone unnoticed in previous manual research, e.g. when investigating vocabulary distribution. In the Perspective Project we use it, for example, in combination with an Ancient Greek corpus that we annotated for speech, attitude and perception reports to see whether there are significant differences between the words a narrator uses when speaking for himself and those in the complements of other people's speeches, thoughts and perceptions. On the other hand, Ancient Greek with its rich morphology and relatively free word order forms an interesting test case for computational linguists to see how natural language processing (NLP) software originally developed for languages that are morphologically poor and have a relatively fixed word order do on languages of a very different kind.

The lemmatization algorithm makes use of Frog [10, 15], an integration of memory-based NLP modules originally developed for Dutch. Frog's contribution to GLEM is two-fold: (i) in the case of words that have more than one lemma in the lexicon and these lemmas have different part-of-speech (POS) tags, GLEM chooses the entry with the POS tag that matches Frog's POS output. (ii) in the case of words that are not already in the lexicon GLEM returns the lemma that Frog generates. Both components will be worked out in section 4.

2 RELATED WORK

One of the first lemmatizers for Ancient Greek was developed in the Perseus project. This lemmatizer originally did not disambiguate: it simply listed all lemmas a word can possibly belong to. Material created in the Perseus project has subsequently been used in various other projects. The Classical Language Toolkit (CLTK) [12] used it to develop a lemmatizer that does disambiguate. This disambiguation is based solely on frequency, though: it simply chooses the most frequently occurring lemma. The lemmatizer created in the Perseus under PhiloLogic project ([1]), which also uses Perseus material, reportedly has disambiguation on the basis of POS information, but this lemmatizer is not publicly available.

As for lemmatization modules more in general, FreeLing [14] offers one that combines a lexicon look up with a set of elaborate and complex affix rules that describe how word forms can be transformed to their root form including deletion and insertion of characters in different positions of the word form. A disadvantage of this lemmatization module is that it is time-consuming for the linguist: the affix rules have to be created by hand, something that we wanted to avoid.

In this respect, GLEM is more akin in spirit to Chrupala's [4] corpus-driven context sensitive approach to lemmatization. Instead of having the linguist write affix rules, the class labels used in this study are based on the same principle of finding those character edits that are needed to transform a word form into its lemma. The labels are represented as *Shortest Edit Scripts*, sequences of character-position insertions and deletions.

As far as we know GLEM is the first publicly available lemmatizer for Ancient Greek that makes use of POS information to disambiguate between multiple lemmas and that also creates lemmas for new words, words not yet in the lexicon, using a trainable memory-based learning component

3 MATERIAL AND PREPROCESSING

We used material from the PROIEL [2] and the Perseus [5] project. The PROIEL project in Oslo created a corpus of Ancient Greek texts (among other languages), annotated for lemma, POS, syntactic analysis and pragmatic information. Its annotated text of Herodotus has been the primary training set for GLEM. It provided us with:

(1) the text of Herodotus with its manual lemma and POS annotations (as far as was finished by February 2015);
(2) the PROIEL lexicon: a lexicon based on (1), with ⟨word, lemma, POS tag⟩ entries, with frequency information;

Since the text of Herodotus is relatively short and written in the somewhat atypical Ionic dialect we used in addition:

(3) the Perseus lexicon, also of the form ⟨word, lemma, POS tag⟩, without frequency information.[1]

We converted (3), which is in betacode (an ASCII-only method of representing Ancient Greek characters, which was the standard for a long time), into unicode using the converter provided by the CLTK.[2] Next, we converted the POS tags, for which the two projects used partly different sets, into a common set of tags and applied this to (1), (2) and (3). We refer to the resulting files as (1'), (2') and (3') respectively. Each POS tag is represented as string of 10 characters specifying grammatical information. For example, the string V--PRPEMG- belonging to the Greek word form προγεγενημένων indicates that is a verb (V), with plural number (p), perfect aspect (r), participle form (p), middle voice (e), masculine gender (m) and genitive case (g). The (1)' version of Herodotus that we used in these experiments contains 90503 tokens, 5445 sentences and 754 different POS-tags.

Finally, we merged the two resulting lexica (2') and (3'), which now use the same set of POS tags, into one list:

(4) our (merged PROIEL-Perseus) lexicon, with ⟨word, lemma, POS tag⟩ entries, without frequency information.

As we will see in the next section, (4) does not replace (2') entirely. Since the lemma disambiguation is partly based on frequency, GLEM consulted (2') for its frequency information in the case of words with multiple lemmas but no POS tag that matches Frog's output.

Furthermore, we annotated a small sample from another Greek author to create a test set that is completely independent from the training material. We took the first 15 chapters from book 1 of Thucydides and manually annotated this with POS tags and lemmas. This final resource that we used contains 2652 tokens, 56 sentences and 314 different POS-tags:

(5) Thucydides test sample

The sample was annotated by three annotators. We computed inter annotator agreement on the first 5 chapters resulting in 92.4% average F-score on the lemmas and 89.2% F-score on the POS tags. We note that F-score is more appropriate than Cohen's kappa for this task, as explained in detail in [11].

GLEM is created under the Creative Commons Attribution- Non-Commercial-ShareAlike 3.0 License and all the material can be found at GitHub.[3] This also includes various rewrite scripts and documentation about mapping the two tag sets into a new one.

4 THE LEMMATIZATION ALGORITHM

GLEM consists of a combination of lexicon look up and Frog. In 4.1 we will first explain the idea behind Frog in general and discuss how we applied it to Ancient Greek. Then, in 4.2, we will explain how GLEM works and uses Frog's outcomes.

4.1 Frog

Frog [10] is an open source natural language processing tool that can be trained for any language with a certain degree of morphology (which degree still being an open question) if there exists a corpus where each word is labeled with its appropriate POS tag and lemma. Frog was originally developed for Dutch and integrates several NLP modules that perform word and sentence boundary detection, POS tag and lemma labeling, named entity recognition and morphological and syntactic analysis. The majority of the modules are based on memory-based machine learning algorithms [6], a technique proposed to learn NLP classification problems with as its defining characteristic that it stores in memory all available instances of a task, and that it extrapolates from the most similar instances in memory to deal with unseen cases.

The Frog POS tagger works as follows. It aims to predict the POS tag for each word in the sentence using information about the word itself and the context around the word like the two preceding words and their predicted POS tag and the following word. The POS tagger consists of two separate sub modules, one to handle words that the tagger has already seen before in its training phase, and a module to handle unseen, new words. The POS tagger already learned for known words which possible POS tags the word can have and uses this information to limit the candidate options. For unknown words, the sub module uses besides the information about the context of the word, also information about the word form itself

[1]Downloadable at http://www.perseus.tufts.edu/hopper/opensource/download.
[2]https://github.com/cltk/cltk/blob/master/cltk/corpus/greek/beta_to_unicode.py.
[3]https://github.com/GreekPerspective.

such as has-initial-capital, first and last letter of the word, contains-hyphen, contains-digit, and last-three-characters. Unknown words are harder to label as the full set of possible POS labels can apply.

The predicted POS tag, together with the characters of the word form, is taken as input by the Frog lemmatizer. Originally developed for Dutch, it is suffix oriented as it focuses on the last 20 characters of the word form. The machine learning classifier learns from examples how it needs to change a word form into its lemma form.

Representing the morphological change from word form to lemma is not trivial as the classifier needs to learn generalizing patterns. We use simple *edit rewrite rules* as class labels that specify which part of the suffix needs to be deleted or inserted to change the word form into its lemma. Let's take as an example the Greek participle verb form βαίνοντες 'going' with lemma βαίνω. The classifier uses a class label that expresses to delete the suffix οντες and then to insert ω. The same rule can be applied to 1274 other verb forms in our Herodotus corpus like στασιάζοντες but also to new unseen verbs with the same ending such as παίζοντές. Note that the classifier takes a very practical approach and changes one surface word form into a canonical lemma form and does not try to model any of the actual underlying morphological rules or historical changes/traces. We refer to the Frog manual [10] for a more detailed description.

We created a light version of Frog for Ancient Greek POS tagging and lemmatizing. We retrained the Frog POS tagger on Herodotus (1'), and we retrained the lemmatization module of Frog on both Herodotus (1') and the merged PROIEL-Perseus lexicon as described in (4).

4.2 GLEM

GLEM makes use of Frog's Greek POS tagger and lemmatizer in the following way (see also Figure 1): First Frog predicts a POS tag for each word form in the text. For each word form and predicted POS tag, a look up in (4), the merged PROIEL-Perseus lexicon, is performed. When a word occurs only once in the lexicon, GLEM simply returns the lexicon's lemma, irrespective of whether there is a POS match or not. When a word form is ambiguous in that it can have multiple possible lemmas, we check whether there exists exactly one form with the same POS tag as predicted by Frog. If it exists, we assign this lemma and are done. If there are multiple forms with the same POS tag or there is no matching POS tag, we use the frequency information from the PROIEL lexicon (2') to choose the lemma of the most frequent word form-POS combination. For unknown word forms that are not part of the lexicon, we use the Frog lemmatizer to generate the most probable lemma form.

As an extra clarification, both Frog and GLEM use both resources (1') and (4), but the main difference is that the Frog lemmatizer uses both resources as training material while GLEM is trying to be more efficient by performing a simple lexicon look up in (4) first and is only falling back to classification for the words that do not occur in the lexicon. Note that the predicted POS tags are leading for the Frog lemmatizer and a wrongly predicted POS tag might result in a wrongly assigned lemma for word forms that are present in the lexicon with another POS tag. Since we have a relative small data set for training the POS tagger for Ancient Greek, and rich fine grained POS tag set, we expect the Frog POS tagger performance to be modest.

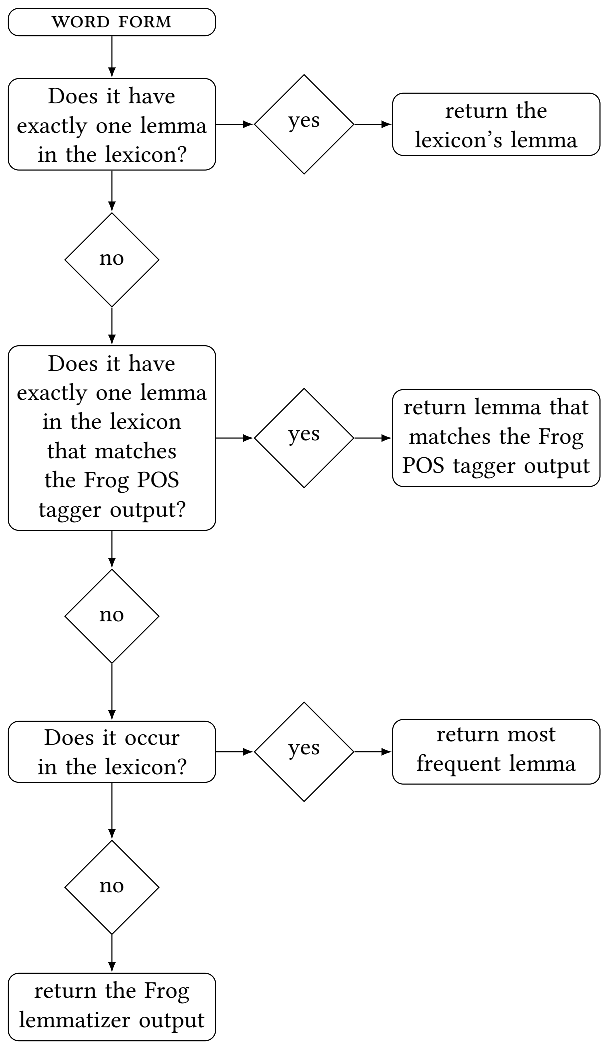
WORD FORM
Does it have exactly one lemma in the lexicon?
yes
return the lexicon's lemma
no
Does it have exactly one lemma in the lexicon that matches the Frog POS tagger output?
yes
return lemma that matches the Frog POS tagger output
no
Does it occur in the lexicon?
yes
return most frequent lemma
no
return the Frog lemmatizer output

Figure 1: Flow chart of GLEM's algorithm

5 EXPERIMENTAL SETUP

We aim to evaluate the quality of the automatically assigned lemmas by both the Frog lemmatizer and our smarter version of it, GLEM. We also compare our results against a current state-of-the-art alternative, the CLTK lemmatizer [12].

We used two different data sets for the evaluation: the corpus of Herodotus (denoted as (1')) and the sample of Thucydides (denoted as (5)) as presented in section 3. We compare the labels predicted by the algorithms against the manually assigned labels.

To evaluate the performance on Herodotus (1'), we ran 10-fold cross validation experiments with both Frog and GLEM on this data set. We also ran an experiment with the CLTK lemmatizer [12] on the Herodotus text as a comparison.

In a second experiment, we used Herodotus (1') as our training material for the POS tagger and lemmatizer of Frog and GLEM, and we used Thucydides (5) as our test sample.

6 EXPERIMENTAL RESULTS

6.1 Results on Herodotus

The accuracy scores of GLEM, the Frog lemmatizer and the CLTK lemmatizer are shown in Table 1. As GLEM incorporated the PROIEL lexicon that was based on Herodotus, we can expect a very high score when we evaluate the lemmatizer on this same data set. Indeed, GLEM achieves an accuracy of 95.7% on lemma prediction in a 10-fold cross-validation experiment. Frog achieves a result of 87.1% on the lemmas. The CLTK lemmatizer achieves a remarkably lower result of 78.7%.

Herodotus ns	lemma accuracy
GLEM (10-cv)	95.7
Frog (10-cv)	87.1
CLTK	78.7

Table 1: Accuracy of the Lemmatizer on Herodotus (1').

One may wonder why GLEM's accuracy isn't even higher. One important reason for this is that (1), the annotated corpus, but not (2), the lexicon based on it, contains #1, #2 etc to disambiguate between senses of lemmas. Since GLEM finds the lemma without # in the lexicon and hence yield this form, all these cases are evaluated as errors since they occur with # in the lexicon. 7 of the top 10 most frequent errors are of this kind.

When we inspected the errors that CLTK made, it turned out that one of the major causes of errors is the definite article. The CLTK lemmatizer returns the form itself rather than the nominal singular masculine form as is done in (1') . For example, the feminine accusative form of the article τὴν has τὴν rather than ὁ as its lemma.

Table 2 gives the results for POS tagging. The POS-tagger of Frog achieves an accuracy of 83.0% in the 10-fold cross-validation experiment on Herodotus. This relatively low score for the task of POS tagging can be easily explained by the fine-grained POS tag set of around 750 different labels and the small training set of 86K tokens. As the Frog lemmatizer relies fully on the predicted POS tag, these POS errors can propagate and harm the performance of the Frog lemmatizer. Interestingly, the GLEM lemmatizer, which is basically the Frog lemmatizer except that if the word occurs uniquely in the lexicon, the GLEM POS tagger replaces the POS tag predicted by Frog by the one in the lexicon, scores 7% higher.

Herodotus ns	POS tag accuracy
GLEM (10-cv)	90.6
Frog (10-cv)	83.0

Table 2: Accuracy of the POS tagger on Herodotus (1').

Returning to lemmatization, we made a detailed analysis for GLEM to discover which tokens were easy or hard to resolve. Around half of the tokens in (1') had only one possible lemma and almost all of these were correctly classified with the lexicon look up. Note that these include around 10% of punctuation marks.

The other 50% of the tokens had multiple entries in the lexicon with different POS tags and these could almost all be resolved (up to 90%) by choosing either the matching POS tag entry or the most frequent entry when no matching POS tag was found.

Only 1.2% of the tokens did not occur in our lexicon and were lemmatized using the Frog module which generated a new lemma form. GLEM succeeded in retrieving the correct lemma with an accuracy of 66.8% (717/1074 cases) for these unknown words.

6.2 Results on Thucydides

The accuracy scores of GLEM, Frog and CLTK on Thucydides are shown in Table 3. As expected, GLEM's accuracy dropped a bit compared to what we found for Herodotus (on which the lemmatizer was trained). This also holds for the CLTK and Frog lemmatizers.

Thucydides	lemma accuracy
GLEM	93.0
Frog	75.6
CLTK	76.6

Table 3: Accuracy of the Lemmatizer on small sample of data taken form Thucydides.

Unsurprisingly, the definite article was still a major cause of errors for the CLTK lemmatizer. The differences for GLEM are more interesting: Whereas in Herodotus only 1.2% were unseen words and hence handled by the Frog lemmatizer, in Thucydides it's around 8%. GLEM resolves 58% of these unknown words correctly. Around 44% of the words in the Thucydides sample has only one lemma and these are almost all (98%) solved correctly. We also inspected the errors made in general but didn't find any clear patterns here. Some errors concerned cases where unknown words were simply resolved incorrectly, in other cases both the predicted and the manually assigned lemma were defendable and simply a different choice was made, and there were even cases where the gold standard was wrong itself.

Table 4 gives the results of the two POS taggers.

Thucydides	POS tag accuracy
GLEM	78.47
Frog	67.5

Table 4: Accuracy of the POS tagger on small sample of data taken form Thucydides.

We inspected the POS errors and found that the following classes of errors contribute to the relatively low score here:

POS tags mapping: the POS tags mapping turned out not to be flawless. Although we intended the mapping to be such that we had a dedicated class of particles, the five most frequent errors are all particles that have been assigned a different POS tag than we intended.

vocative: the Ancient Greek vocative case is almost always the same form as the nominative case. Since nominatives are much more frequent than vocatives this ambiguity should almost always be resolved in the favor of the nominative. For words that were not in (1), however, and for which we therefore neither have frequency information nor a pattern of how it occurs in a text, we often find the vocative form wrongly predicted.

7 CONCLUSIONS

The comparison with the CLTK and Frog lemmatizer suggests that for Ancient Greek indeed a combination of lexicon look up and memory based learning, as forms the idea behind GLEM, works better than a lemmatizer which uses only one of the two. We noticed that due to the small training sample in combination with the rich POS tag set, the performance of the Frog POS tagger and the Frog lemmatizer, which depends heavily on the predicted POS tags, were not optimal and only achieved an accuracy around 83 and 87%, respectively. GLEM was developed to overcome this issue and the combination of look up and Frog in this case indeed turned out to yield a higher accuracy.

ACKNOWLEDGMENTS

This research is supported by the EU under FP7, ERC Starting Grant 338421-Perspective. We thank Kees Thijs, Leopold Hess and Dag Haug for their help.

REFERENCES

[1] 2016. Perseus under PhiloLogic. http://perseus.uchicago.edu. (2016). [Online; accessed Dec 20, 2016].
[2] 2016. PROIEL: Pragmatic Resources in Old Indo-European Languages. https://github.com/proiel/, http://www.hf.uio.no/ifikk/english/research/projects/proiel/. (2016). [Online; accessed Dec 23, 2016].
[3] 2016. Thesaurus Linguae Graecae. http://stephanus.tlg.uci.edu. (2016). [Online; accessed Dec 23, 2016].
[4] Grzegorz Chrupa la. 2006. Simple data-driven context-sensitive lemmatization. *Procesamiento del Lenguaje Natural* 37 (2006), 121–127.
[5] Gregory R. Crane. 2016. Perseus Digital Library. http://www.perseus.tufts.edu. (2016). [Online; accessed Dec 16, 2016].
[6] Walter Daelemans and Antal van den Bosch. 2010. Memory-based learning. In *Computational Linguistics and Natural Language Processing Handbook*, Alex Clark, Chris Fox, and Shalom Lappin (Eds.). Wiley-Blackwell, 154–179.
[7] Helma Dik and Richard Whaling. 2008. Mining Classical Greek Gender. Paper presented at Digital Humanities, Oulu, Finland, June 25–29. (2008).
[8] Dag T. Haug, Marius L. Jøhndal, Hanne M. Eckhoff, Eirik Welo, Mari J. B. Hertzenberg, and Angelika Müth. 2009. Computational and Linguistic Issues in Designing a Syntactically Annotated Parallel Corpus of Indo-European Languages. *Traitement Automatique des Langues (TAL)* 50, 2 (2009), 17–45.
[9] Dag T. T. Haug and Marius Jøhndal. 2008. Creating a parallel treebank of the old Indo-European Bible translations. In *Proceedings of the Language Technology for Cultural Heritage Data Workshop (LaTeCH 2008), Marrakech, Morocco, 1st June 2008*. 27–34.
[10] Iris Hendrickx, Antal van den Bosch, Maarten van Gompel, and Ko van der Sloot. 2016. *Frog, A Natural Language Processing Suite for Dutch, Reference guide.* Technical Report Draft 0.13.1. Radboud University Nijmegen. Language and Speech Technology Technical Report Series 16-02, available at https://github.com/LanguageMachines/frog/raw/master/docs/frogmanual.pdf.
[11] George Hripcsak and Adam S. Rothschild. 2005. Agreement, the F-Measure, and Reliability in Information Retrieval. *Journal of the American Medical Informatics Association (JAMIA)* 12, 3 (2005), 296–298.
[12] Kyle P. Johnson and others. 2014–2016. CLTK: The Classical Languages Toolkit. https://github.com/cltk/cltk. (2014–2016). [Online; accessed Nov 12, 2016].
[13] Francesco Mambrini. 2016. Treebanking in the world of Thucydides. Linguistic annotation for the Hellespont Project. *DHQ: Digital Humanities Quarterly* 10, 2 (2016).
[14] Lluís Padró and Evgeny Stanilovsky. 2012. FreeLing 3.0: Towards Wider Multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*. ELRA, Istanbul, Turkey.
[15] Antal van den Bosch, Bertjan Busser, Sander Canisius, and Walter Daelemans. 2007. An efficient memory-based morpho-syntactic tagger and parser for Dutch. In *Computational Linguistics in the Netherlands*, P. Dirix, I. Schuurman, V. Vandeghinste, and F. Van Eynde (Eds.). 99–114.