Progress in machine parsing of ancient Greek

Benjamin Crowell
Istituto universitario di Fullerton,
emeritus

Machine lemmatization and part-of-speech tagging of ancient Greek have been done using multiple methods, including pattern matching and unsupervised machine learning. The accuracy of the results has generally been much lower than for other European languages. I describe significantly improved results obtained with a large lookup table of inflected words, built using multiple sources of lexical data. For lemmatization of Attic prose, its failure rate is about an order of magnitude lower than those of other existing models. Testing shows that when attempting to resolve ambiguities in the part of speech, no existing model does much better than a strategy of frequency-based guessing plus a few simple pattern-matching heuristics to take advantage of context.

1. Introduction

Systematic lemmatization and part-of-speech tagging of ancient Greek goes back at least 170 years, to Wigram's hand-constructed Analytical Greek Lexicon (Wigram 1852), in which every inflected form in the New Testament was listed alphabetically with its POS analysis and lemma. An early computerized attack on the problem was begun in 1985 by Joshua Kosman and Neel Smith, and later completed and described in the literature by Gregory Crane (Crane 1992). This system, called Morpheus, was adapted to the computing resources of the time, and was also limited by the nonexistence of the internet or publicly available lexical resources in machine-readable form, so that the authors were compelled to construct their own lexical database by hand. The system, which remains in use today, uses pattern-matching techniques, although these are difficult to apply to Greek, not just because of its complex morphology but because of the language's movable accent, which, as pointed out by Smith (Smith 2016), makes certain classes of parsing algorithms inapplicable.

Thus machine inflection of the language is an easier problem than machine parsing, but the parsing problem can be reduced to that of inflection, provided that one has sufficient computing resources. In the work described here, a large lookup table was constructed which contained every possible inflected form in the language — or, in practice, a big enough set of forms so that other, rarer ones could be reduced to them through pattern-matching.

Although the current trend in morphological parsing of high-resource languages is to use character-level neural network models, we will see that, compared to the method presented here, that approach performs badly on ancient Greek.

2. Morphological Engine

The system is based on a morphological engine that represents a Greek word in a format more convenient than its direct representation as a unicode string. For example, "Μῆνιν" would be represented as the two-element list (*μη*,*νιν*), consisting of one Unicode string for each syllable, the syllabification being carried out according to a modernized and rationalized version of the best practices of ancient scribes (Stuart-Jones 1901). Additional meta-data describe the length of the morae; the word's accentuation, capitalization, and any breathing marks; and whether the word is a compound, contains crasis or elision, or is an incomplete word such as a stem or suffix. An object in this format can be converted to and from a canonical Unicode form using the NFC normalization with only composed characters.

The engine, named Ifthimos, includes facilities for concatenation and pattern matching, which are considerably complicated by the need to take into account the morphological processes of sandhi and vowel contraction. Since these processes are sometimes optional or unpredictable, or may depend on data such as etymology that are not available, an operation such as concatenation is in general a many-valued function, and its software implementation returns a list of results rather than just one word. This one-to-many design of the API is an important factor in making the results of the computations trustworthy; in comparison, most neural-network approaches to natural language parsing are designed so that they always produce exactly one result, which may not be the right one and may in some cases be a hallucination. In addition, all operations in the system described here produce an "explainer" along with the list of results. The explainer is a list of strings that explain how the result was arrived at, for example by referring to a section number in one of the standard grammars. The name Ifthimos is meant to evoke the fact that the software was designed from the ground up to provide results that can be trusted, or at least verified in the sense that a human can tell how they were obtained.

An additional benefit of incorporating contraction and sandhi in all operations is that, compared to earlier systems, it allows the morphological production rules to be stated in simpler form and with fewer exceptions, and the number of distinct categories of declension and conjugation patterns is greatly reduced.

A disadvantage of this design is that it can have an impact on computing time. When a series of operations is carried out, each of which produces multiple results, the result is a decision tree whose number of nodes grows exponentially. This problem is somewhat mitigated because the depth of the tree is not too great for real words, and the software also uses some heuristics to prune the tree. However, the computational requirements would have helped to make such a design computationally impractical in the era of (Crane 1992). An especially time-consuming operation turns out to be the parsing of compounds, since Greek has historically tended to load more and more prepositions onto the front of verbs, and the morphological rules for stringing them together are complex.

The full set of morphological operations includes many for which one needs to know the POS and lexical data such as the lengths of vowels in a stem. The POS tag, if known, is stored along with the word. The software can carry out most operations with incomplete data. For example, if told to construct a word object from the written form *σωτῆρα*, it does not require a lexicon to tell it that the final alpha is short, since that can be inferred from the accent. However, lexical data are taken advantage of in some cases when they are available.

3. Sources of Lexical Data

Lexical data were compiled from two main groups of sources. Dictionary entries were extracted from Wiktionary and from the Liddell-Scott-Jones dictionary (Henry George Liddell and Stuart-Jones 1843), the latter with ad hoc pattern-recognition scripts to extract machine-friendly data. In addition, I compiled lemma and POS data for Greek texts from multiple treebanks: the Perseus Digital Library's Ancient Greek Dependency Treebank (AGDT) (Project Perseus 2021), the Global Bible Initiative (Global Bible Initiative 2024), and Zeman and Swanson's treebank of the Septuagint version of Genesis.(Zeman and Swanson 2024) Some other treebanks were collected in the same repository but excluded from the analysis because their licenses were not compatible with the others or with the intended license of this project.

The treebanks are in theory the best training data, because they tell us how words were actually inflected by ancient authors. However, they contain a large number of errors, which has been acknowledged to be a serious problem by workers using them as training data (Dik and Whaling 2009). Problems also arise due to differences in the tagging habits of the individuals who participated in the collaborations (Giuseppe Celano and Majidi 2016). The AGDT is no longer actively maintained, with the result that multiple researchers have created their own versions of it. The most notable prior systematic attempt to correct errors has been in a proprietary system at the University of Chicago (Dik and Whaling 2008). As part of the present work, I have constructed a database of the POS and lemmatization data from the multiple sources listed above, and have tried to correct as many errors as possible and to make more uniform the tagging habits of the individual workers. To my knowledge this is the only such systematic effort that has been made publicly accessible and has respected the intention of the treebanks' original creators to make them part of the public commons without further restrictions.

In the business of classification, "lumpers" and "splitters" have always been in conflict. For example, the verbs *διαρριπτέω* and *διαρρίπτω* (to scatter) are variants of one another, and their forms could be lemmatized either separately or together, according to taste. The treebank collaborators sometimes lumped and sometimes split. But for the purpose of training a machine to inflect words, splitting such variants is desirable, and that is the policy that I have tried to impose in my patches to the databases.

4. Analysis and Synthesis

The software then automatically determined the stems and inflection classes from the tagged data. This was done not by unsupervised machine learning using neural networks but by strict algorithmic analysis in terms of human-defined inflection classes. For example, the noun *ἔγχος* (spear) occurs in eleven forms in the treebank, all of them classified as neuter nouns. In addition to the lexical form, we have the plural nominative *ἔγχεα*. These two observations are sufficient to determine that the word is a third-declension neuter noun of an inflection class that is labeled in the software as 3-n-os. The other nine inflected forms are also consistent with this analysis. Therefore the word is recorded as having stem *ἔγχ-* and this inflection class.

Verbs are treated in much the same way, except that each principal part is handled separately. Ancillary information is gathered in a separate data structure, so that, for example, *εἶπον* is tagged as a second aorist.

These computations take about 50 hours of CPU time, so that on current desktop hardware with parallel processing, they can be completed in a few hours of wall-clock time.

As a supplement to this analysis, additional data on stems and inflection classes were extracted from Wiktionary and Liddell-Scott-Jones (LSJ), the latter being especially useful for rare lemmas. In some cases principal parts of verbs were inferred algorithmically, but some trial and error was needed in order tell which patterns were actually reliable. For example, if one were to be given only the dictionary head-word *πιέζω*, it would be possible to infer reliably that it would have a first perfect *πεπίεκα*, but for a word like *τάσσω* one cannot infer the perfect *τέταχα* without knowing further information, such as the etymology.

Once the stems and inflection classes have been determined, a comparable amount of CPU time is required in order to build a look-up table of all possible inflected forms. The resulting database is about 4 Gb in size and contains 22M inflected forms.

5. Parsing

Once the database has been constructed, parsing is in principle very simple. A given word has its accentuation and capitalization converted to a canonical form. We then look up that form in the database. This may result in multiple hits, since, e.g., *φύλλα* (leaves) can be either nominative or accusative. Better results can, however, be obtained by supplementing the table lookup with a little bit of pattern-matching.

Since formation of compounds from verbs is productive, one can always encounter forms that are not in the lexical data, e.g., a complex compound such as *ἐξυπανίστημι* (=*ἐξ*-*ὑπό*-*ἀνά*-*ἵστημι*). This would be reduced by pattern matching to a simpler form such as *ἀνίστημι* or *ἵστημι* for which lexical data are available. Pattern matching is also used for crasis, and for suffixes found in words like *οἵοισπερ*, *τουτουί*, and *τοσοίδε*. The code also uses pattern matching on verbal adjectives in -*τέος*, some compound numerals such as *τετρακισχίλια*, and for rare parts of speech such as the future perfect participle, which is only known to have occurred once in all of classical Greek.

The treebanks are overwhelmingly slanted toward a few dialects: epic, Attic, and koine. In the lookup table, we attempt to produce correct forms for all of these, plus Ionic. Each word is recorded in the lookup table along with a four-bit mask indicating which of these "big four" dialects it could occur in. Queries to the parser specify a potentially complex data structure giving a fine-grained description of the dialect and period, but when the search is actually carried out this is reduced (possibly with a loss of precision) to a four-bit mask. Some dialectical differences are handled by pattern-based manipulation of the input to make it more "vanilla." For example, Attic *θάλαττα* would be automatically converted to *θάλασσα*, and Ionic *ἀπίκετο* to *ἀφίκετο*.

When the fancier pattern-matching is turned off, the parser is rather fast. For example, the 10M word Diorisis corpus (Vatri and McGillivray 2018) was lemmatized in 6 hours of wall-clock time on a 16-core 5.4 GHz Intel desktop machine, taking advantage of memoization. This converts to about 35 ms per word. With full functionality, including POS disambiguation as described below, the CPU time per word is about double this figure.

6. Tests of Performance

6.1 Failure Rate for Lemmatization

To evaluate the performance of this model for lemmatization, I tested it along with two NN systems, Stanza 1.7.0 (Qi et al. 2020) and OdyCy 0.7.0 (Kostkan and Kardos 2024), which I chose because they have parsers trained on ancient Greek and are freely available under licenses that are compatible with common licenses for free and open-source software as defined by the Open Source Initiative. In addition, I tested Morpheus 1.0.0 (September 15, 2017). All three main parsing approaches were therefore represented: pattern-matching, table lookup, and NN. The software to reproduce all the tests is publicly available (sec. 7).

I measured the success of each parser in lemmatizing words in one Attic text and one in the epic dialect. In order to avoid bias, I used texts that were not included in any of the codes' training data: for Attic, the *Cyropaedia* by Xenophon, and for the epic dialect, book 13 of the *Astronautilia*, by Jan Křesadlo. The *Astronautilia* is a modern science fiction epic written in an imitation of the Homeric style. Because the NN models often hallucinate nonexistent lemmas, I only counted the output as a success if it was one of the head-words in the Liddell-Scott-Jones dictionary (or an equivalent when filtered through a simple function to account for differences in spelling, dialect, and the differing choices of lexicographers). The *Astronautilia* contains many words that are Křesadlo's own coinages, and when these words occur in the text, all the algorithms are almost guaranteed to fail, since the (correct) lemma will not have been in LSJ. One therefore should take the results on that text only as a comparison of the models, not as an absolute measure of their performance. As inputs, I excluded the 3000 most common words in ancient Greek.

In this test, I did not require that the software produce the *right* lemma. For example, if the software output a list of possibilities, that was considered a success. It would have been more desirable to test whether each model got the lemmatization right according to some ground truth. Such a notion is slippery, however, since it depends so much on the preferences of a particular lexicographer, such as whether to use active *οἴω* or middle *οἴομαι* as a lemma.

The results of this test are shown in Table 1. For standard Attic, the failure rate of the model described here is an order of magnitude less than those of the other models.

Table 1
Failure rates (percentages) in the test of ability to lemmatize words, excluding the 3000 most common words. Lemming is the parser described in this work.

	Morpheus	Lemming	Stanza	OdyCy
Cyropaedia (Attic)	25	3	18	30
Astronautilia (epic)	39	26	27	47

6.2 Disambiguation of the Part of Speech

The algorithm as described so far considers words entirely in isolation, so that if the POS is ambiguous based on the written form, it is impossible to disambiguate it except on a statistical basis, such as by knowing that although the vocative case usually has the same form as the nominative, the nominative is far more common. Neural network (NN) methods for POS tagging are designed to take advantage of context, and therefore could potentially do better at this task, although this is somewhat of an apples-to-oranges comparison, since the nature of the application determines whether one cares more about getting a correct and complete list of possibilities or a single educated guess. It is of interest, however, to know whether NN models *do* in fact succeed at using context for this task in Greek. They were after all designed, tested, and optimized for English.

Greek's highly inflected nature and its relatively free word order make it a very different test case.

I am concerned here only with the fine-grained portion of the POS tags, such as case and tense, not with the coarse POS categories such as preposition and adverb, represented by the first character in the Perseus tagging system. Previous work (Giuseppe Celano and Majidi 2016) shows that the latter depends so greatly on lexicographical preferences as to be difficult to evaluate. To the extent that I have made an effort to impose more uniformity on such choices in my patches to the training data, testing the ability of parsers to reproduce the coarse-grained POS would merely verify that my own parser behaved according to my own preferences.

I formulated two tests of whether the NN models made effective use of context. These were constructed as one-on-one comparisons between the parser described in this work (Lemming) and one of the NN models. The test data came from the first chapters of Xenophon's Cyropaedia and Hellenica, which have been tagged by hand for POS by Vanessa Gorman (Gorman 2024).

In the first testing method, I looked for words for which Lemming and the NN model agreed on a single lemma, and Lemming found more than one possible POS analysis. I took the list of possible analyses output by Lemming for a given word, and selected the most likely one based on a hand-coded heuristic, which is essentially an estimate of frequency, with smoothing to handle the incomplete coverage of the set of fine-grained POS analyses. For example, the nominative and vocative cases usually look the same, but the nominative is more common, so in such cases it would be selected as the more likely one. My test code would then count how often this probabilistic guessing method was right and how often the NN model was right (assuming the human analysis to be correct). I computed the ratio r of the former count divided by the latter. A result of $r = 1$ would support the extreme null hypothesis that the NN model made no effective use of context at all. Such a result would say that the model's output was like predicting the weather in Southern California simply by saying that every day will be sunny.

The value of r was 0.87 for Stanza and 0.82 for OdyCy. That is, 82-87% of the time, the disambiguation of an ambiguous POS could have been accomplished equally well by guessing the most common POS.

In the second test, I randomly shuffled the order of the words in each sentence. If this had little effect on the model's accuracy, it would be evidence that the model was not making much effective use of "local" context, i.e., what a word's nearby neighbors were in the sentence. I found that shuffling the order of the words in a sentence caused very little degradation in the performance of Stanza. Without shuffling, Stanza correctly disambiguated the part of speech 87% of the time. With shuffling, it did it 83% of the time. Thus Stanza does make some successful use of local context for disambiguating POS, but the extent of this success is limited. The figures for OdyCy were 88 and 81%, showing somewhat more effective use of context.

I was able to improve Lemming's frequency-based heuristic to make a little use of context and give results that came even closer to matching the NN methods. To do this, I implemented a few simple pattern-matching algorithms in order to do things like correlating the gender and number of an article with the gender and number of the nominal that it refers to. For example, suppose that the text contains the phrase *τοῖς στρατηγοῖς*. The gender of *τοῖς*, taken in isolation, can be either masculine or neuter. The word *στρατηγοῖς* is masculine and agrees with *τοῖς* in case and number, so we guess that the gender of *τοῖς* is likely to be masculine. With several such improvements, the value of the r statistic became 0.91 for Stanza and 0.89 for Odycy.

It seems, then, that the NN models tested here do not make very much effective use of context to disambiguate the POS analysis of a language with the specific characteristics of ancient Greek. As an indication of the progress that remains to be made on this front, all the systems tested here fail to correctly parse the sentence *φύλλα μῆλα ἐσθίουσιν*, "sheep eat leaves." The difficulty is that there are no surface clues such as inflection or word order that distinguish the role of the subject and object in this sentence, so that the models are apt to interpret it as "leaves eat sheep," or to assign both words the same case, which is syntactically impossible. The correct parsing of this type of ambiguity in natural language requires data about semantics or lexical preferences (Hindle and Rooth 1993). The trend in the field has been to turn away from attempts to encode such data explicitly and instead to try to train NN models so that they incorporate them implicitly, but the models tested here do not seem to have succeeded at this for ancient Greek. The language's corpus is not very big and not growing, so one cannot simply throw more data at the NN models. Some workers in the field have been making attempts to build a body of more explicit semantic data that would be relevant for this type of task, for example in an ancient Greek implementation of WordNet (Boschetti, Del Gratta, and Diakoff 2016).

7. Implementation and Applications

The model is implemented in about 45,000 lines of Ruby code, and the database in SQLite. (When interfacing with Python code, I have either set up a pipeline that passed data through text files, or else executed one language from within the other using a shell.) The code is under the GPL open source license, and the data are either public domain or under the licenses chosen by the third parties who did the original work. The code can be obtained from `https://bitbucket.org/ben-crowell/lemming/src/master/INSTALL.md`. The code to run the performance comparisons in section 6 is available at `https://bitbucket.org/ben-crowell/test_lemmatizers/src/master/summary.md`.

The model has been used in several applications to date. I have used it to create a student text of Xenophon's Anabasis with glosses based on machine-generated lemmatization; in an online Greek Word Explainer application, which explains the inflectional rules used to produce a given Greek word; for lemmatization as a preprocessing step in the creation of `word2vec` word embeddings (Tomas Mikolov and Dean 2013) from the Diorisis corpus; and as part of an English-Greek bitext alignment system.

8. Conclusions

This paper has described a new system for lemma and POS tagging of ancient Greek using table lookup. When used on standard Attic Greek, its failure rate as a lemmatizer is about an order of magnitude lower than those of other presently existing systems that use pattern matching or neural network approaches. It enumerates all possible lemma-POS analyses and provides a human-readable description of how it arrived at them.

Although the main finding of this work is that an explicit algorithm works much better for this task than NN methods, there are also various ways in which the two approaches could complement one another or be used together. A NN lemmatizer could be constrained to produce a result from among the possibilities enumerated by Lemming. NN output could be supplemented with human-readable explanations from this code. Unsupervised machine learning models would probably perform better if they were trained on the data described in section 3, which have had some errors corrected and lumping/splitting decisions made with a view to such applications.

Near the completion of this work, I got in touch with Neel Smith, one of the original authors of the Morpheus code, and learned that he was working on a similar project using table lookup for parsing. I thank him for his cordial and helpful communication. His forthcoming system promises to have better handling of dialectical differences than the one described here, and will also cover Latin and Hebrew.

References

Boschetti, Federico, Riccardo Del Gratta, and Harry Diakoff. 2016. Open ancient Greek WordNet 0.5. ILC-CNR for CLARIN-IT repository hosted at Institute for Computational Linguistics “A. Zampolli,” National Research Council, in Pisa.

Crane, Gregory. 1992. Generating and parsing classical Greek. *Literary and Linguistic Computing*, 6(4):243–245.

Dik, Helma and Richard Whaling. 2008. Bootstrapping classical Greek morphology. *Digital Humanities 2008*, pages 105–106.

Dik, Helma and Richard Whaling. 2009. Implementing Greek morphology. *Digital Humanities 2009*, pages 338–339.

Giuseppe Celano, Gregory Crane and Saeed Majidi. 2016. Part of speech tagging for ancient Greek. *Open Linguistics*, pages 393–399.

Global Bible Initiative. 2024. Greek New Testament. `https://github.com/biblicalhumanities/greek-new-testament`, accessed January 6, 2025.

Gorman, Vanessa. 2024. Greek dependency treebanks. `https://github.com/rgorman/Greek_Dependency_Treebanks/blob/master/README.md`, accessed January 6, 2025.

Henry George Liddell, Robert Scott and Henry Stuart-Jones. 1843. *A Greek-English Lexicon.* Oxford University Press.

Hindle, Donald and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational linguistics*, 19(1):103–120.

Kostkan, Jan and Márton Kardos. 2024. Odycy. `https://github.com/centre-for-humanities-computing/odyCy`, accessed October 2, 2024.

Project Perseus. 2021. Ancient Greek dependency treebank. `https://github.com/PerseusDL`, accessed December 5, 2023.

Qi, Peng, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages.

Smith, Neel. 2016. Morphological analysis of historical languages. *Bulletin of the Institute of Classical Studies*, 59(2):89–102.

Stuart-Jones, Henry. 1901. The division of syllables in Greek. *Classical Review*, 15(8):396–401.

Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 3781.

Vatri, Alessandro and Barbara McGillivray. 2018. The Diorisis ancient Greek corpus. *Research Data Journal for the Humanities and Social Sciences*, 3(1):55–65.

Wigram, George. 1852. *Analytical Greek Lexicon of the New Testament.* Bagster and Sons.

Zeman, Dan and Daniel Swanson. 2024. UD ancient Greek PTNK. `https://github.com/UniversalDependencies/UD_Ancient_Greek-PTNK`, accessed August 11, 2024.