MACHINE LEARNING FOR ANCIENT GREEK LINGUISTICS

A TENTATIVE METHODOLOGY AND APPLICATION TO THE CORPUS OF DOCUMENTARY PAPYRI

Master's thesis submitted in partial fulfilment of the requirements for the degree of

MASTER OF ARTS IN DE TAAL- EN LETTERKUNDE

Submitted by: Xavier Goas Aguililla
Supervisor: dr. T. Van Hal
KU Leuven
Faculty of Arts
Department of Greek Studies

Leuven, 2013

Xavier Goas Aguililla: *Machine learning for ancient Greek linguistics,* © August 2013.
Typeset with pdfTeX 3.1415926-2.4-1.40.13 using a modified version of Lorenzo Pantieri's ArsClassica package.

E-MAIL:
xavier.goas@student.kuleuven.be

ABSTRACT

SAMENVATTING

Deze masterproef bevat **61371** tekens.

PREFACE

ACKNOWLEDGEMENTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 THESIS STATEMENT

The following work is concerned with the application of techniques from the field of artificial intelligence and machine learning to an ancient Greek corpus, namely that of the documentary papyri.

It intends to prove that it is possible to use these techniques, which are generally applied to English and other modern languages, to generate linguistic annotation for this corpus in an accurate and efficient manner.

In order to achieve this, I have implemented the architecture proposed in Collobert & Weston 2008 and Collobert et al. 2011. This architecture relies on cutting-edge techniques in machine learning and offers the possibility of using data devoid of linguistic annotation as part of the training corpus. After implementing this architecture, I have taken a sample from fully tagged texts and used it to check the general accuracy of the system. I have done the same for the papyri, albeit manually and using a smaller sample due to the fact that there is no preexisting annotation for the corpus of documentary papyri.

1.2 MOTIVATION

The study of the language of the papyri has in the past thirty years seen little evolution until the recent appearance of Evans and Obbink's *The Language of the Papyri* [Evans and Obbink, 2010], which has placed the subject in the spotlight again. Twentieth-century scholarship on the topic, though still useful for those interested in the study of the papyri for historical purposes, is either antiquated, limited in scope or incomplete (see *infra*). Despite this, the papyri are useful source material for the history and evolution of the Greek language, as they contain not only official texts but private documents as well, whose linguistic features and peculiarities have the potential to foster new insights into the nature of colloquial Greek. Such a corpus could be a boon to scholars interested in the Greek of the papyri, as it would facilitate, for instance, the creation of linguistically sound grammars and lexica.

Furthermore, applying the same methodology to other Greek texts offers new linguistic perspectives. The largest corpus of ancient Greek, the Thesaurus Linguae Graecae, contains more than a hundred million words. A system is in place which offers morphological and lemmatic analysis in a rudimentary form, but this cannot serve as a full linguistic corpus. Using manual methods to tag this corpus is rather prohibitive

due to its size, but using computational methods, we can make an attempt at offering a relatively complete linguistic corpus for ancient Greek.

I also wish to demonstrate the potential of a methodology based on computational techniques for the study of the classical languages. The field has seen a move towards digitalisation in the past half century, but a lot of potential is left untapped. Steps in the right direction are currently being made, but we can drive progress much farther. The classicist breed does not number many specimens; and though the true classics, the Homers, the Platos, the Virgils, have been subjected to thorough analysis for millennia, the amount of texts in need of scholarly attention remains large. Demonstrating the potential of computational methods for Greek linguistics will hopefully serve as further proof of their potential for other branches of classics, such as stylometry, authorship verification, textual criticism, and more.

1.3 GOALS AND OBJECTIVES

1.4 CONTRIBUTIONS

1.5 OUTLINE

I begin by giving an overview of the background for this thesis. First the historical and linguistic background of the question is handled, expounding on previous efforts to study the grammar of the papyri and to apply the techniques of corpus linguistics to the Greek language in general. Second comes a technical section, describing a set of core concepts and techniques which are relevant to the task at hand and are necessary to understand the underpinnings of the applied methodology.

This is followed by an analysis of the objectives and requirements, as well as a general (that is, only described in broad strokes, without providing full details of the algorithms and implementation) methodology. Correspondingly, a chapter is dedicated to the general algorithmic design and structure of our program, followed by a chapter delving into the actual implementation, which provides more details on the choice of programming language, the availability of the source code, the technical requirements for running our code, etc.

We then offer an interpretation and a critical assessment of our program's output. Special attention is, of course, given to the linguistic accuracy of our results. A final chapter is dedicated to an overview of further possibilities for research in this field and of possible applications outside papyrology and to the field of Greek linguistics in general.

2 BACKGROUND

2.1 HISTORICAL BACKGROUND

2.1.1 The language of the papyri

The papyri began to be studied linguistically not by papyrologists and historians, but rather by Bible scholars and grammarians interested in their relevance in the development began to koinê Greek, particularly that of the New Testament. G. N. Hatzidakis, W. Crönert, K. Dieterich, A. Deissmann, and A. Thumb pioneered the field in the late nineteenth and early twentieth century, spurring a resurgence of scholarship on the topic;[1] an excellent overview of pre-1970s research may be found in Mandilaras, 1973 and Gignac, 1976, 1981a.

During this period, E. Mayser begon work on the earliest compendious grammar of the papyri; it limits itself to the Ptolemaic era but explores it at length and in great detail. The work [Mayser, 1938] consists of a part on phonology and morphology, made up of three slimmer volumes, and a part on syntax, encompassing three larger volumes. Its composition seems to have been exhausting: it took Mayser thirty-six years to finish volumes I.2 through II.3, with I.1 only completed in 1970 by Hans Schmoll, at which point the entire series was given a second edition.

When casually browsing through some of its chapters (though casual is hardly the word one would associate with the *Grammatik*) it is remarkable to see that Mayser brings an abundance of material to the table for each grammatical observation he makes, however small it may be. For instance, the section on diminutives essentially consists of pages upon pages of examples categorised by their endings.

This is its great strength as a reference work - whenever one is faced with an unusual grammatical phenomenon in any papyrus, consulting Mayser is bound to clarify the matter; or rather, it was, for the work is now inevitably dated. The volumes published during Mayser's lifetime only include papyri up to their date of publication; only the first tome by Schmoll includes papyri up to 1968. It is still a largely useful resource, but it is in urgent need of refreshment.

After Mayser set the standard for the Ptolemaic papyri, a grammar of the post-Ptolemaic papyri was the new *desideratum* in papyrology. The work had been embarked on by Salonius, Ljungvik, Kapsomenos, and Palmer, only to be interrupted or thwarted by circumstance or lack of resources. Salonius, 1927, for instance, only managed to write an introduction on the sources, though he offered valuable comments on the matter of deciding how close to spoken language a piece of

1 Vide Crönert, 1903; Deissmann, 1895, 1897, 1929; Dieterich, 1898; Thumb, 1901, 1906.

writing is. Ljungvik, 1932 contains select studies on some points of syntax.

It is in the 1930's that we see attempts to create a grammar of the papyri that would be the equivalent of Mayser for the post-Ptolemaic period. S. Kapsomenos published a series of critical notes [Kapsomenos, 1938, 1957] on the subject; though he attempted at a work on the scale of the *Grammatik*, he found the resources sorely lacking, as the existing editions of papyrus texts could not form the basis for a systematic grammatical study. The other was L. Palmer, who had embarked on similar project and had already set out a methodology [Palmer, 1934]; the war interrupted his efforts, and he published what he had already completed, a treatise on the suffixes in word formation [Palmer, 1945].

A new work of some magnitude presents itself two decades later with B. G. Mandilaras' *The verb in the Greek non-literary papyri* [Mandilaras, 1973]. Though it does not aim to be a grammar of the papyri, it does offer a thorough and satisfactory treatment of the verbal system as manifest in the papyri. Further efforts essentially do not appear until the publication of Gignac's grammar. It is essentially treading in the footsteps of Mayser, only with further methodological refinement and a more limited, though still sufficiently exhaustive, array of examples. The author, for reasons unknown to me, only managed to complete two of the three projected volumes, on phonology and on morphology. The volume on syntax is thus absent, a gap only partly filled by Mandilaras' *The verb in the Greek non-literary papyri*.

Finally, there is the aforementioned *The Language of the Papyri* [Evans and Obbink, 2010], which does not aim to be a work on the same scale as the aforementioned. It is a collection of articles on various topics, the whole of which is meant to illuminate new avenues for future research. A particularly relevant chapter for this thesis is the last one by Porter and O'Donnell [Porter and O'Donnell, 2010], who set out to create a linguistic corpus for a selection of papyri; their tagging approach, however, is manual, and their target corpus limited. The authors also are the creators of `http://www.opentext.org/`, a project aiming for the development of annotated Greek corpora and tools to analyse them; sadly, no progress seems to have been made since 2005.

2.1.2 Corpus linguistics

A[2] corpus or text corpus is a large, structured collection of texts designed for the statistical testing of linguistic hypotheses. The core methodological concepts of this mode of analysis may be found in the concordance, a tool first created by biblical scholars in the Middle Ages as an aid in exegesis. Among literary scholars, the concordance also enjoyed use, although to a lesser degree; the eighteenth century saw the creation of a concordance to Shakespeare.

The development of the concordance into the modern corpus was not primarily driven by the methods of biblical and literary scholars;

2 The following section is based *passim* on McCarthy and O'Keeffe [2010].

rather, lexicography and pre-Chomskyan structural linguistics played a crucial role.

Samuel Johnson created his famous comprehensive dictionary of English by means of a manually composed corpus consisting of countless slips of paper detailing contemporary usage. A similar method was used in the 1880s for the Oxford English Dictionary project - a staggering three million slips formed the basis from which the dictionary was compiled.

1950s American structuralist linguistics was the other prong of progress; its heralding of linguistic data as a central given in the study of language supported by the ancient method of searching and indexing ensures its proponents may be called the forerunners of corpus linguistics.

Computer-generated concordances make their appearance in the late 1950s, initially relying on the clunky tools of the day - punch cards. A notable example is the Index Thomisticus, a concordance to the works of Thomas of Aquino created by the late Roberto Busa S.J. which only saw completion after thirty years of hard work; the printed version spans 56 volumes and is a testament to the diligence and industry of its author. The 1970s brought strides forward in technology, with the creation of computerised systems to replace catalogue indexing cards, a change that greatly benefited bibliography and archivistics.

It is only in the 1980s and 1990s that are marked the arrival of fully developed corpora in the modern sense of the word; for though the basic concepts of corpus linguistics were already widely used, they could not be applied on a large scale without the adequate tools. The rise of the desktop computer and the Internet as well as the seemingly ever-rising pace of technological development ensured the accessibility of digital tools. The old tools - punch cards, mainframes, tape recorders and the like - were gladly cast aside in favour of the new data carriers.

The perpetual increase of computing power equally demonstrated the limits of large-scale corpora; while lexicographical projects that had as their purpose to document the greatest number of possible usages could keep increasing the size of their corpora, the size of others went down as they whittled the data down to a specific set of uses of language.

The possible applications of the techniques of corpus linguistics are diverse and numerous; for they allow for a radical enlargement in scope while remaining empirical, and remove arduous manual labour from the equation. Corpus linguistics can be an end to itself; it can, however, assert an important role in broader research. McCarthy and O'Keeffe, 2010, p. 7 mention areas such language teaching and learning, discourse analysis, literary stylistics, forensic linguistics, pragmatics, speech technology, sociolinguistics and health communication, among others.

The term 'corpus' has a slightly different usage in classical philology: they designate a structured collection of texts, but that collection is not primarily intended for the testing of linguistic hypotheses. Instead, we have, for instance, the ancient corpus Tibullianum, or modern-day

collection, for instance the Corpus Papyrorum Judaicarum, etc. We are primarily interested in the digital techniques used to create linguistic corpora; so let us first take a look at the progress of the digital classics.

2.1.3 The digital classics

Classical philology, despite its status as one of the oldest and most conservative scientific disciplines still in existence today, has in the last fifty years found itself at the front lines of the digital humanities movement. Incipient efforts in the fifties and sixties, mainly stylometric and lexical studies and the development of concordances, demonstrated the relevance of informatics in the classics, an evolution that was at first met with some skepticism, but later fully embraced.

The efforts began with the aforementioned Index Thomisticus, the first computer-based corpus in a classical language; but the first true impetus was the foundation of the Thesaurus Linguae Graecae project in 1972, a monumental project with as its goal the stocking of all Greek texts from the Homeric epics to the fall of Constantinople. Over the years, many functions have been added to this ever more powerful tool; and even in the beginning stages of its development, the TLG garnered praise.

The usefulness of the tool in its current form cannot be overstated: not only does it contain a well-formatted and easily accessible gigantic collection of text editions whose scope and dimensions exceed those of nearly any university library; it also offers all of these texts in a format that allows for lexical, morphological and proximity searches, as well as including a full version of the Liddell & Scott and Lewis & Short dictionaries. The TLG has become a staple of the digital classics.

Despite this, the TLG is becoming more and more dated as technology progresses. While recent years have seen the rise of Unicode as the standard for encoding ancient Greek, the TLG still uses beta code, a transliteration system designed to only use the ASCII character set, and the texts are stored using an obsolete text-streaming format from 1974, which divides the text in blocks of eight kilobytes and marks the division between segments.

A digitised version of the Liddell-Scott-Jones lexicon has been added to the TLG's web interface, but the texts themselves have not undergone extensive tagging, only lemmatisation. Searching through the database can be done by searching for specific forms of a lemma, or by searching for all forms of a lemma, but this is essentially the limit of the search tool's power; it is not possible to perform a query for all possible lemmata associated with a particular form, i.e. we cannot find all forms which are, for example, an active perfect indicative.

In the wake of the TLG, several notable projects have emerged: Brepols' Library of Latin Texts is trying hard to be for Latin texts what the TLG is for Greek texts; the Packard Humanities Institute has released CD's containing a selection of classical Latin works. In more recent times, the Perseus Project has enjoyed great popularity because of the attractive combination of an excellent selection of classical texts with transla-

tions, good accessibility and a set of interesting textual tools, the entire package carrying a very interesting price tag for the average user — it is free to use, and for the greatest part, open source as well.

The databases I have mentioned are quite general in scope; but within the domain of classical philology, other specialised projects exist. Within the field of papyrology, for instance, the digital revolution has taken a firm foothold. Starting with several separate databases, the field has experienced a tendency towards convergence and integration of the available resources, as exemplarised by the papyri.info website, maintained by Columbia University, that integrates the main papyrological databases into a single database.

A great feature of this database is the shell in which all data is wrapped; they are compliant with the EpiDoc standard, a subset of XML based on the TEI standard and developed specifically for epigraphical and papyrological texts. One may access the database's resources through the Papyrological Navigator and suggest corrections and readings through the Papyrological Editor. What's more, all data is freely accessible under the Creative Commons License, crowd-sourced, regularly updated, and can be downloaded for easier searching and tweaking.

In other words, papyri.info has brought the open-source mentality from the computer world into the classics. For our purposes, this open setup is desirable, as the database is not fit for them as it is, but can with some effort be molded into a useful tool.

2.1.4 Natural language processing

Natural language processing (henceforth NLP) is a subdiscipline in computer science concerned with the interaction between natural human language and computers. Its history well and truly starts in the fifties, with a basic concept which has played a great role in natural language processing, and computer science in general, the Turing test. This test, put forth by Alan Turing in his seminal paper *Computing Machinery and Intelligence* [Turing, 1950], evaluates whether a machine is intelligent or not by placing a human in conversation with another human and a machine; if the first human cannot tell the other human and the machine apart, the machine passes the test.

Machine translation systems entered development, though progress soon stalled because of technical limitations and because of methodological obstacles: such systems were dependent on complex rulesets written by programmers that allowed for very little flexibility. Because of the slow return on investments made, funding for artificial intelligence in general and machine translation specifically was drastically reduced throughout the late sixties and the seventies.

A resurgence followed: thanks to advances in computational power and the decline of Chomskyan linguistics in natural language processing, which had been the dominant theoretical vantage point in the preceding thirty years, the eighties were marked by the introduction of statistical machine translation, which is fundamentally based on the

tenets of corpus linguistics. Modern natural language processing is therefore situated on the crossroads between various fields: artificial intelligence, computer science, statistics, and corpus and computational linguistics. It looks to be an exciting field for the coming years as its techniques are under constant improvement and ever more present in our daily lives.

Most NLP software is designed explicitly with living languages in mind; English, being a world language and the international *lingua franca*, has enjoyed most of the attention, but other major languages have enjoyed some attention, too. Ancient languages, however, are neglected, presumably due to their often high complexity and the extensive study and analysis to which they have been submitted by skilled scholars. Yet most texts have not been integrated in annotated corpora; and though databases such as the Perseus project contain large swathes of morphologically and sometimes syntactically annotated text, the process has been driven largely by manual labour; to give an exhaustive list is not appropriate here, but another such example which is relevant is the PROIEL project [*PROIEL: Pragmatic Resources in Old Indo-European Languages*], which is also a treebank, i.e. a database of syntactically annotated sentences. It contains data for Herodotus and the New Testament.

2.2 CONCEPTS AND TECHNIQUES

While there is, of course, no room in this thesis for an extended course in mathematics or computer science, it is necessary to have some background in order to understand the techniques used for the design and implementation of the architecture. What follows is an brief overview of important concepts and techniques from mathematics, artificial intelligence and natural language processing applied in this thesis. For the less mathematically inclined reader, formalism has been reduced to a minimum; instead, for each concept and technique, only a layman's explanation is given. For further formalism, see A on page A-1.

2.3 RELATED WORK

Computational approaches to classical philology have been the object of increasing interest for the last few years. While none have chosen to focus on the language of the Greek papyri specifically, related areas have received attention and are relevant to the task at hand. Annotated corpora have been created, efforts to automatically tag Greek have been made, and some have even taken a stab at using natural language processing techniques for textual criticism.

2.3.1 Artificial intelligence techniques applied to ancient Greek linguistics

H. Dik and R. Whaling

In two papers based off their workshops on the topic [Dik and Whaling, 2008, 2009], H. Dik and R. Whaling (a classics professor and computer scientist turned classicist, respectively, both at the University of Chicago) demonstrate a relatively simple methodology for morphological tagging of a corpus of ancient Greek in the context of the Perseus under PhiloLogic project under Helma Dik. They trained Helmut Schmid's TreeTagger (found at `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/` and extensively described in Schmid, 1994, 1995) using a corpus of Homeric and New Testament Greek and applied it to run over their three-million word corpus. Initial results achieved about 80% accuracy and rose up to 88%; since the corpus was designed for academic use, large swathes of text were then manually disambiguated to the best of the ability of the authors and a team of volunteers.

Their effort is remarkable in the sense that it is the only instance of an automatically annotated corpus of ancient Greek I have managed to find. While Perseus offers a morphological analysis tool, this tool is designed to assist linear reading and generates parses on the fly from a database, offering several options if several parses are possible. Perseus does not consider the context; Dik and Whaling's corpus, though, has been annotated with the explicit intention of doing so and then storing all parses and their context in a relational database. This makes it possible to perform morphological searches, which, for linguistic purposes, is very interesting.

J. Lee

Lee, 2008 offers a new approach to the analysis of Greek morphology by using machine learning methods. The method proposed relies on large amounts of data and makes use of nearest-neighbour analysis techniques. This stands in contrast to the traditional rule-driven approach used by earlier parsers such as Packard, 1973 or Crane, 1991.

By applying affix transformations to word forms in order to analyse their relation to other forms, a nearest-neighbour metric is established on the basis of the number of these transformations needed to generate another extant word form. An annotated corpus is fed to the training architecture supported by a large amount of unlabelled data to facilitate the prediction of verbal stems.

An accuracy of 98.2% is achieved for words present in the training set, with the remaining forms necessitating contextual disambiguation. For words not present in the training set, an average accuracy of 85.7% is achieved with most of the loss in accuracy due to words with most inaccuracies due to what the author terms 'novel roots', which are stems which are not directly derivable from a word form and are analysed with a practical 50% accuracy (though this was improved to 65% by loosening the standards by which accuracy was measured).

The use of machine learning techniques in this paper is laudable and certainly not badly executed, but the general methodology and in particular the choice of training corpora is problematic. An annotated version of the first five books Septuagint is used to establish these metrics. This corpus consists of 470K words and can be reduced to a set of about 37K unique words. Yet it is unnecessary to restrict training to such a corpus if one does not look at n-grams but only at stand-alone word forms. The question, then, is: why not use the output of Morpheus, the system designed by Gregory Crane and honed over the years, as training material? It is freely available as an SQL database and contains ~1M unique parses which are generated using very large corpora for verification. The nearest-neighbor metric may be applied to these word forms and arguably form a better picture, as well as allowing for better extrapolation to data not present in the training corpus.

Despite this glaring flaw, the applied method is still laudable in the sense that it shows that it is not necessary to create very complex rule-sets for morphological analysis. With minimal integration of linguistic knowledge, we can create a simple heuristic which offers reasonable results.

Lee and D. Haug, 2010

2.3.2 Collaborative / open-source philology

D. Bamman

Bamman and Crane, 2008, 2009, 2011; Bamman, Mambrini *et al.*, 2009; Bamman, Passarotti *et al.*, 2008

G. Crane; the Open Philology Project

3 ANALYSIS

3.1 OBJECTIVES

Our main objective is the creation of a **statistical language model** for ancient Greek using techniques from machine learning; as an illustration, we will then apply it to the **corpus of papyri** as provided by papyri.info.

Such a model, at heart, serves to assign probabilities to word sequences; however, it is key that the model is also able to apply the knowledge of these probabilities to specific problems in the analysis of language. The problems we want to tackle are classical **morphological analysis** and **syntactic analysis**.

The first problem largely corresponds with what is called **part-of-speech tagging** in the natural language processing jargon. For any token in the sentence, given its context, we want to model to produce a morphological analysis, which produces not only the part-of-speech, but all concomitant information as well: voice, tense, mood, case, gender, number, person, ...

The second problem corresponds with what is called **shallow parsing** (or **chunking**). Given a sequence of words, we want to identify the main grammatical components of this sequence. This type of parsing stands in contrast with **deep parsing**, which aims to produce full syntactic analyses of entire sentences.

3.2 SCOPE OF THE PROBLEM

We summarise the major component problems of our task facing us.

A first major obstacle is the size and diversity of the corpus. The corpus of the papyri as presented on papyri.info contains more than 4.5 million words and spans nearly a millennium, by virtue of which it inevitably contains tens of thousands of unorthodox or corrupted word forms. This adds a great amount of complexity: not only must our model recognise 'normal' word forms, it must also be able to make inferences (albeit limited ones) about unknown forms with minimal hiccups.

- quantity: roughly 4.5 million words
- quality: varying from extremely corrupted text to nearly pristine documents
- representation: from 300BC to 800AD, representing an array of different discourse registers, but not including literary texts

- simplicity: offered in EpiDoc XML, which is more complex than plain text, but is relatively easily converted
- retrievability: extensive annotation using XML makes searching and retrieving text easy

The second major obstacle is the ancient Greek language itself. Though it has lost a great deal of morphological complexity in its evolution towards its current state, the Greek of Hellenistic, Roman and Byzantine times is still marginally more complex than a language like English, which is the target language for most research in natural language processing. Commonly used techniques in NLP are still applicable and have been used with success on other morphologically complex languages, but given the size of the tag set for ancient Greek morphology and syntax, it is wise to preprocess the corpus to reduce the amount of factors that must be held into account in the creation of our model.

Another important obstacle to the development of natural language processing tools for ancient Greek is the relative scarcity of training material: most resources do provide morphological analysis, but there are very few projects concerned with treebanks or databases of semantically annotated Greek. The Perseus project, for instance, has developed a dependency treebank for Latin and ancient Greek. It is an admirable effort, but limited in scope and containing mainly poetry, which is in itself valid training data, but certainly not sufficient training data if we want our system to be able to analyse large amounts of prose. What's more, the project seems to be lacking manpower and has lost steam since its inception, the last update dates from 2012, more than a year ago at the time of this writing.

Another interesting treebank is that hosted by the PROIEL [1] project, which aims to offer morphologically and syntactically annotated multilingual corpora for comparative purposes. The project, contrary to the Perseus treebank, seems to be alive and well at the time of this writing. This corpus contains data which can be of much help: large swathes of Herodotus, the New Testament, and the writings of the Byzantine historian George Sphrantzes are fully annotated, both morphologically and syntactically. Since the Greek of the papyri is syntactically similar to the Greek of these texts, we are afforded a good basis for our system.

Nevertheless, probabilistic natural language processing is by virtue of its underlying principles hungry for ever more data in order to achieve high performance. Therefore, we somehow need to create a larger foundation upon which we can construct a performant architecture. Here, we can take our cues from the field of machine learning, where the state-of-the-art approaches rely on massive amounts of unlabeled data which are then submitted to analysis. The exact approaches chosen are explained in detail in the next chapter.

1 Short for 'Pragmatic Resources for Indo-European Languages'

4 DESIGN

4.1 INSPIRATION

The idea of designing a system to automatically process ancient Greek was originally inspired by the work of R. Whaling and H. Dik, who used a purely supervised method for tagging a corpus of classical Greek. They trained H. Schmid's TreeTagger using a relatively small corpus of annotated Greek and proceeded to tag their corpus with it, offsetting the relatively large error rate with manual work done by graduate students at the University of Chicago. Whaling and Dik's method enabled them to annotate the corpus in much less time than would have been necessary would the annotation process have been executed manually.

An early prototype of this thesis attempted to use similar supervised methods to annotate the corpus of the papyri. Despite high expectations, experience showed that the lack of extensive annotated corpora is a severe hindrance, as the main way to improve the accuracy of any NLP system is to offer it more training data. Feeding 400.000 words as training data to the Stanford POS Tagger resulted in a measly 60% accuracy on a validation set held out from the training corpus.

Recent literature in the field revealed that state-of-the-art results were being achieved using a combination of unsupervised and supervised learning techniques, dubbed semi-supervised approaches. Unsupervised approaches can make use of unannotated data as a preparation for supervised training, and work by trying to divide the raw data into clusters on the basis of various criteria. Notably, R. Collobert and others developed a versatile architecture which achieved high accuracy on several NLP tasks and required a relatively low amount of optimisation. See Collobert and Weston [2008], Collobert, Weston *et al.* [2011]. Accuracy for POS tagging reached up to 97.20%, while for chunking, scores of up to 93.63% were achieved; similarly high results were achieved for named entity recognition and semantic role labeling. this is an impressive performance given the fact that most of the architecture is shared among all tasks and the majority of the parameters of the system are inferred through unsupervised methods.

Given that far larger amounts of raw textual material are available for ancient Greek, it seems that this kind of technique is suited to the problem at hand. The 400.000 word training corpus used in the experiment with the Stanford POS Tagger is much smaller and limited than corpora like that offered by the Perseus project (about 7M words) and the TLG (about 109M words at last count, though these are not freely available). Making use of this untapped resource is desirable. This chapter is dedicated to an overview of the architecture; the approach

followed in Collobert, Weston *et al.* [2011] and Turian *et al.* [2010] is respected with amendments and simplifications where needed in order to accommodate for some characteristics of Greek (in particular the very high complexity of its morphology requires a subtler approach). The exact implementation of the system is left for the next chapter.

4.2 CREATING A LANGUAGE MODEL

Neural networks as a technique are anything but new; the general concepts underlying them date from more than half a century ago, and thousands of applications have been found that make use of them. A single-layer neural network is a compact structure that can perform complex tasks efficiently; and even with the hardware which was in use a decade ago and before, training a neural network was a feasible task. Training a deep neural network, which contains several hidden layers (hence the term *deep*), is another matter; doing this requires the backpropagation algorithm, which until the mid-2000's was simply too slow for use on the hardware of the day.

At this point, G. Hinton, who had been one of the first proponents of the use of deep networks in the 1980's as a professor at Carnegie Mellon University, blew new life into the field he helped create with his paper "Learning multiple layers of representation" [G. E. Hinton, 2007]. He demonstrated that it was possible to perform very complex learning tasks relatively efficiently and to great effect using this type of network. The way this is done is by stepping down from the traditional approach where neural networks are given a certain amount of classified examples and training them to classify observations based on these; rather, the goal is now to train **generative networks**, i. e., networks that can randomly generate possible observations. Examples given to the network do not need to be classified in advance; instead, given an observation, the network's parameters are adjusted to maximise the likelihood of data of its kind being generated.

A classic task for this type of network is handwritten digit recognition; a network is trained by feeding it a large amount of examples of handwritten digits. For each example, the parameters are adjusted; after a sufficiently large amount of examples, the network is capable of generating handwritten digits itself, in a vast number of variations.

Applying the method to natural language processing requires the development of a structured internal representation for each word (or each letter, but we will consider words). If we choose to designate each word by a fixed-length vector with n components, these vectors define how the word is embedded within an n-dimensional vector space; hence, these vectors are termed **embeddings**.[1]

The unified architecture proposed by Collobert et al. uses these embeddings, which are tailored during training using a huge amount of natural language data, to initialise networks which will be trained in

1 To avoid confusion: note that the components of this feature vector do not necessarily show a one-on-one correspondence with linguistic features!

a supervised manner. These networks operate in the same manner and make use of the same embeddings, with the difference that they are task-specific and trained on classified examples. Performance improvements are due to the fact that at this stage, most of the general learning is actually already done and we are applying classification to certain clusters in the vector space, which allows the model to make more accurate inferences when classifying rare words or phrases.

4.2.1 Network structure

Deep networks contain multiple layers which are sequentially trained; the input of a layer is weighted and passed on to the following layer, which may be either an output layer or a hidden layer. Several layers are stacked in this manner.

Hyperparameters

Before constructing a network, we need to decide on a set of hyperparameters that will influence the parameters of the neural network and its training process. These are:

- the **embedding dimensions**, written d_{wrd}: the number of components in a word feature vector;
- the **dictionary size**, written D: how many words we want to consider when training;
- the **window size**, written *wsz*: the number of words we want to consider per example;
- the **learning rate**, written λ: when using gradient descent, how much we want to adjust the network parameters at each gradient step;
- the **embedding learning rate**, written λ_e: the same as λ, but for the purpose of learning embeddings;
- the **input, output and hidden size**, written n^l_{in}, n^l_{out}, and n^l_{hu}, respectively: the number of neurons contained in a layer l.

Lookup table layer

The lookup table itself is a matrix with d_{wrd} rows and D columns. Its initial construction is as follows: given a frequency table, the D most common words are chosen and placed in order. Each word is now assigned an index according to its ranking in the frequency table. The most common word gets index 1, the second most common index 2 etc., up to the word in the D^{th} place in the table, which is assigned index D. For each index, a new embedding of size d_{wrd} with small random values is created and assigned to that index. The lookup table matrix itself is constructed by concatenating these D vectors as

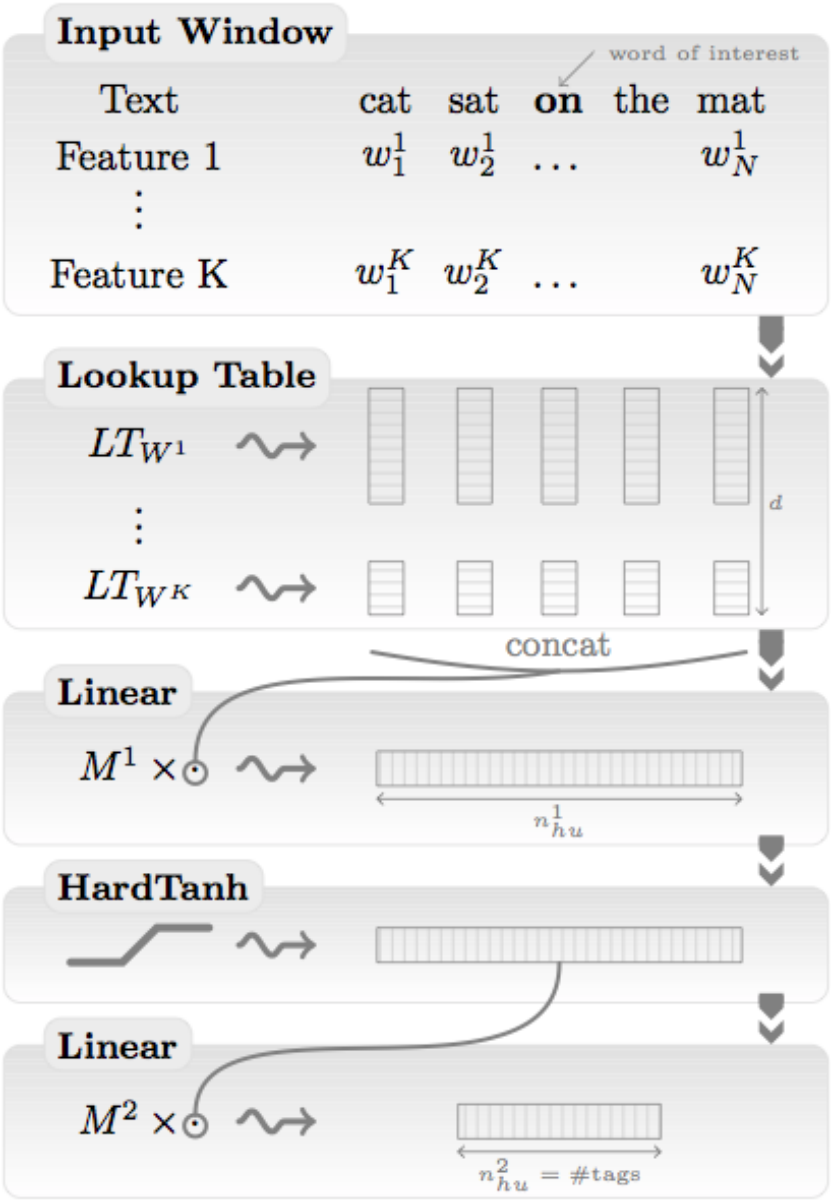

Figure 1: The basic network structure, using a window approach. Figure from Collobert, Weston *et al.*, 2011, p. 2499.

columns. In this way, a lookup operation for an index i is actually nothing more than the selection of the i^{th} column from this matrix.

The lookup table layer itself consists of wsz input neurons; given a text window, each word is converted to its corresponding index, which is then fed to the neuron corresponding to its position in the window, which retrieves the embedding of that word from the lookup table. The output of all neurons is then concatenated into a single matrix, whose columns are now the embeddings for the input window.

Formally, given a word index vector $w \in \mathbb{N}^{wsz}$ and a lookup table $L \in \mathbb{R}^{d_{wrd} \times D}$, we can express this as as a function LT:

$$LT(L, w) = \begin{pmatrix} L_{1,w_1} & L_{1,w_2} & \cdots & L_{1,w_{wsz}} \\ L_{2,w_1} & L_{2,w_2} & \cdots & L_{2,w_{wsz}} \\ \cdots & \cdots & \cdots & \cdots \\ L_{d_{wrd},w_1} & L_{d_{wrd},w_2} & \cdots & L_{d_{wrd},w_{wsz}} \end{pmatrix} \tag{1}$$

Linear layer

A linear layer takes a fixed size vector and performs a linear operation on it: the dot product of this vector with a set of parameters W is computed and a bias added. Formally, given the output vector f_θ^{l-1} of layer $l-1$, the following computation is performed in layer l:

$$f_\theta^l = W^l \cdot f_\theta^{l-1} + b^l \tag{2}$$

Where θ indicates the existing parameters of the network and $W^l \in \mathbb{R}^{n_{hu}^l \times n_{hu}^{l-1}}$ and $b^l \in \mathbb{R}^{n_{hu}^l}$ are the parameters of the layer to be trained, with n_{hu}^l representing the amount of hidden units in layer l. Linear layers transform their input and several such layers can be stacked, similar to how linear functions can be composed.

Hard hyperbolic tangent layer

If we intend for our network to be able to model a highly nonlinear system such as language, we need to introduce nonlinearity somewhere. A good function for this is the hyperbolic tangent, which is differentiable everywhere and approximates a linear threshold function very nicely. A layer l using the hyperbolic tangent as an activation function contains n_{hu}^l neurons taking n_{hu}^{l-1} inputs. In this case, the activation function $g(x)$ for a scalar x is:

$$g(x) = tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{3}$$

For an input vector generated by a layer $l-1$, the function g represented by a hyperbolic tangent layer can be defined as:

$$f_\theta^l = g(f_\theta^{l-1}) = \begin{bmatrix} g(f_{\theta 1}^l) \\ g(f_{\theta 2}^l) \\ \cdots \\ g(f_{\theta n_{hu}^{l-1}}^l) \end{bmatrix} \tag{4}$$

We approximate this function using the hard hyperbolic tangent, defined for a scalar as:

$$\mathtt{hardtanh}(x) == \begin{cases} 1, & \text{if } x > 1 \\ -1, & \text{if } x < -1 \\ x & \text{otherwise.} \end{cases} \tag{5}$$

We can define this function for vector inputs in the same manner as for the hyperbolic tangent function.

Scoring layer

The final layer. This is actually simply a linear layer which is designed to output a vector containing as many elements as there are possible tags for the task at hand. Each output element is a score which reflects the probability of the corresponding tag for the central word in the input window.

4.2.2 Unsupervised learning

The first phase of learning is unsupervised; large amounts of raw language data are fed to the network. Instead of training using a classical squared-loss function, a pairwise ranking function is introduced. The network is constructed as described in the previous section; we want a single score $f_\theta(x)$ to be output for a given window of text x. The window is first corrupted using a word w from the dictionary by replacing the central word in x by w. We express this corrupted window as $x^{(w)}$. The pairwise ranking of any two such pairs x and w is defined as $r(\theta, x, w) = \max\left\{0, 1 - f_\theta(x) + f_\theta(x^{(w)})\right\}$. In effect, we want the non-corrupted window to achieve a higher score than the corrupted window. We can achieve this by adjusting the parameters θ such that the pairwise ranking of x and w is minimal, since this implies that $f_\theta(x)$ must yield a higher score than $f_\theta(x^{(w)})$. Summing this operation over all possible pairs (x, w) and defining a mapping from the parameters θ to this sum, we obtain a general cost function:

$$\theta \mapsto \sum_{x \in X} \sum_{w \in D} r(\theta, x, w) \tag{6}$$

Where X is the set of possible windows of size wsz and D the chosen dictionary. Minimizing this function with respect to θ will ensure the relevant parameters (the embeddings and the first two layers) are tuned so that our ranking function f_θ yields accurate scores. Using this simple criterion, we have a method for crafting a set of parameters that contains a consistent structured internal representation of the training data. Despite the relatively simplicity of the criterion, the huge amounts of parameters results in a very taxing and lengthy computation. Furthermore, there is no guarantee that the cost function

has a single minimum with respect to θ; a full grid search would be necessary, which necessitates vast amounts of computing time.

Instead, the process is sped up using **curriculum learning**; the basic idea of this technique is analogous to the learning process children are put through in school: instead of starting their education with university-level quantities of difficult material immediately, a restricted set of elementary concepts is introduced on which they concentrate. Successive phases of learning are performed by gradually expanding the set of concepts which is to be learned, making use of earlier concepts to facilitate the understanding of more complex concepts.

The same method, for reasons not yet fully understood, can be applied to unsupervised learning.[2] First, the training material is restricted to the most frequent observations of the process we want to model. Training over this restricted set creates a simplified model which, due to the abundance of examples, should be accurate. Subsequently, new iterations of the learning algorithm are run over successively larger sets; at each iteration, the model becomes more detailed and describes more classes of observations more accurately.

This is applied to the problem at hand by choosing successively larger dictionary sizes. During the calculation of the minimum of the cost function, windows which are not centered around a word which is in the dictionary are ignored. This initially entails a significant reduction of our sets X and D, which makes the process a bit less computationally demanding. Subsequent iterations are computationally more expensive, but are initialised with the parameters found by previous iterations; observations previously used in learning will already return excellent scores and will only necessitate minor adjustments to the parameters, while new observations can be fit into the general picture more easily.

4.2.3 Supervised learning

The supervised training phase involves the creation of task-specific networks which are initialised with the embeddings created during the unsupervised phase; these form a shared first part of all networks. A given network is then tailored to have an adapted output as necessary for the task of interest. Training proceeds in the classical fashion, by providing correctly formatted examples and adjusting all parameters (including the shared ones) to minimize the squared error.

The output of a network of this type is a vector; each specific feature is encoded in on component of this vector by setting this component to one and all others to zero. This is called a one-hot vector. All tasks are jointly trained, that is to say, the networks share parameters, are trained simultaneously and are allowed to modify shared parameters. Individual (non-shared) network parameters are modified only during training for a specific task. This technique allows us to generalise the training benefit from each example.

2 Among the scholarship of note on this subject we find Bengio, Louradour *et al.* [2009] and Erhan *et al.* [2010].

Composing such a one-hot vector is not simple for Greek morphology: any given word will belong to multiple morphological categories. For instance, a verbal form has a tense, a mood, a number, a person, and sometimes even a case and gender. Gathering all possible features into a single one-hot vector is therefore not feasible.

We could approach this problem in different ways. An instinctive test is to simply feed the system raw parses and assign one component of the output vector to each possible parse. This is needlessly complicated; if we look at the list of morphological parses from the Perseus database, we find more than 2000 distinct morphological analyses! An output vector of this size is simply too unwieldy.

A more dynamic approach would be to create one-hot vectors for each of the following categories, with the number of options assigned to each category corresponding to the number of components in the corresponding output vector:

- major part of speech: verb, noun, adjective, pronoun, particle , adverb, numeral, preposition, conjunction, interjection;
- minor part of speech: article / determinative, personal, demonstrative, indefinite, interrogative, relative, possessive, reflexive, reciprocal, proper;
- person: first, second, third;
- number: singular, plural, dual;
- tense: present, imperfect, aorist, perfect, pluperfect, future, future perfect;
- mood: indicative, subjunctive, optative, imperative, infinitive, participle, gerundive, gerund, supine;
- voice: active, middle, passive, middle-passive;
- gender: masculine, feminine, neuter, common;
- case: nominative, genitive, dative, accusative, ablative, vocative;
- degree: comparative, superlative.

This approach has its downside: we now have to train distinct networks for each network. The upside, though, is that each of these networks is much, much smaller than a single network mapping all possible parses and will be easier to train; an example of the divide-and-conquer technique. We could possibly be confronted with impossible parses, such as an 'imperfect optative', but this is highly unlikely due to the total absence of examples for this form.

The task of preprocessing the corpus, which is discussed *infra*, is also simplified due to this approach: the two main corpora from which the training material is gathered use similar but slightly different annotation schemes. The PROIEL system is a bit more complex, but essentially gives the same information as the Perseus system. Tables 1 and 2

field	category	possible values
first field	lemma type	a - adjective c - conjunction d - adverb e - exclamation g - particle l - article m - numerals n - noun p - pronoun r - preposition t - participle v - verb x - miscellanea
second field	person	1 - first person 2 - second person 3 - third person
third field	number	d - dual p - plural s - singular
fourth field	tense	g - gerund p - participle a - aorist f - future i - imperfect l - pluperfect p - present r - perfect t - future perfect
fifth field	mood	i - indicative m - imperative n - infinitive o - optative s - subjunctive
sixth field	diathesis	a - active e - energetic m - medial p - passive
seventh field	gender	f - feminine m - masculine n - neuter o, p and q - unclear
eighth field	case	a - accusative d - dative g - genitive n - nominative v - vocative
ninth field	degree of comparison	c - comparative s - superlative

Table 1: The Perseus ennealiteral morphological abbreviation system.

field	category	possible values
first field	person	1 - first person 2 - second person 3 - third person
second field	number	d - dual p - plural s - singular
third field	tense	a - aorist f - future i - imperfect l - pluperfect p - present r - perfect t - future perfect
fourth field	mood	i - indicative m - imperative n - infinitive o - optative s - subjunctive
fifth field	diathesis	a - active e - energetic m - medial p - passive
sixth field	gender	f - feminine m - masculine n - neuter
seventh field	case	a - accusative d - dative g - genitive n - nominative v - vocative
eighth field	degree of comparison	c - comparative s - superlative
ninth field	placeholder column	-
tenth field	inflectibility	i - inflected n - not inflected

Table 2: The PROIEL decaliteral morphological abbreviation system.

field	value	Perseus first field equivalent
A-	adjective	→ a
C-	paratactic conjunctions	→ c
Df	adverbs	→ d
Dq	adverbial response particles (where, how, etc.)	→ g
Du	adverbial question particles (where, how, etc.)	→ g
F-	Hebrew loan words	→ x
G-	hypotactic conjunctions	→ c
I-	illocutive particles	→ g
Ma	cardinal numerals	→ m
Mo	ordinal numerals	→ m
Nb	nouns (in general)	→ n
Ne	nouns (proper names)	→ n
Pc	pronouns (reciprocative)	→ p
Pd	pronouns (demonstrative)	→ p
Pi	pronouns (interrogative)	→ p
Pk	pronouns (reflexive)	→ p
Pp	pronouns (personal)	→ p
Pr	pronouns (relative)	→ p
Ps	pronouns (possessive)	→ p
Px	pronouns (quantitative, i.e. some, all, none, same, other)	→ x
R-	prepositions	→ r
S-	article	→ l
V-	verb	→ v

Table 3: The PROIEL biliteral lemmatic abbreviation system.

The Perseus system	The PROIEL system
• adv: adverbial; • apos: apposing element; • atr: attributive; • atv/atvv: complement; • auxc: conjunction; • auxg: bracketing punctuation; • auxk: terminal punctuation; • auxp: preposition; • auxv: auxiliary verb; • auxx: commas; • auxy: sentence adverbials; • auxz: emphasizing particles; • coord: coordinator; • exd: ellipsis; • obj: object; • ocomp object complement; • pnom: predicate nominal; • pred: predicate; • sbj: subject.	• adnom: adnominal; • adv: adverbial; • ag: agens; • apos: apposition; • arg: argument (object or oblique); • atr: attribute; • aux: auxiliary; • comp: complement; • expl: expletive; • narg: adnominal argument; • nonsub: non-subject (object, oblique or adverbial); • obj: object; • obl: oblique; • parpred: parenthetical predication; • part: partitive; • per: peripheral (oblique or adverbial); • pid: Predicate identity; • pred: predicate; • rel: apposition or attribute; • sub: subject; • voc: vocative; • xadv: open adverbial complement; • xobj: open objective complement; • xsub: external subject.

Table 4: The Perseus and PROIEL syntactic annotation systems.

offer a detailed overview of both systems; table 3 contains conversion guidelines. For simplicity's sake, we pick the Perseus system, since this limits the amount of networks we'd have to train to nine. Instead of having to convert from one type of annotation to another, we can simply split the annotations into their constituent parts, store them in different files and resolve any annotational differences afterwards with minimal headaches.

The process of dependency annotation only requires one network, but is a bit more complex. The source corpora for the training data once again use similar annotation schemes but with different emphases. We enumerate the tags for the Perseus and the PROIEL annotation system, respectively.

We see in table 4 that they respectively use nineteen and twenty-four different features. The Perseus system is less detailed than the PROIEL system, but by virtue of this fact also less complex. A possible approach is to map overlaps in both systems and find the simplest possible tag set which can be derived from that. We can immediately make a one-hot vector from this, since all annotated words are equipped with a single syntactig tag.

4.3 SELECTING AND PROCESSING TRAINING DATA

For the unsupervised learning phase, we need a maximally large corpus. I chose the TLG CD-ROM E, which contains about 9.3M words, and the Perseus texts, which contain about 7.7M words. Since both corpora shared material, duplicated sentences were scrapped. The final corpus contains about 16.9M words. For training the model, this corpus is split sentence-wise into a training corpus, from which representations are learned, and a validation corpus, to check the accuracy of the generated representations. The file is split 90-10.

The supervised learning phase makes use of the Perseus treebank and the PROIEL annotated texts of Herodotus and the New Testament. These respectively contain approx. 350K and 195K words, making for a total of about 545K words. Again, a validation set is withheld, in a slightly lower proportion than in the unsupervised phase due to the restricted size of the corpus.

All texts are preprocessed by converting all words to lowercase and placing exactly one sentence on every line. Only Greek punctuation is left; critical notation etc. is pruned.

4.3.1 Possible improvements

4.4 ANNOTATING THE CORPUS

After the architecture (model and networks) is built, it is serialised; that is to say, the internal state of the architecture during training time is

stored to disk. Serialisation allows us to immediately load the model into memory during the execution of our tagging program. When tagging, we use the architecture in a read-only manner, i.e. we only predict and do not adjust parameters any more.

Tagging is essentially a process of probabilistic prediction; text windows are passed through each of the networks, which return a prediction of the expected features of the central words in these windows in the form of an output vector. For tagging a sentence with n words, we create n text windows and use these as input. Each window generates an output vector; we pick the component with the maximum score and attribute the corresponding tag to the central word in the original text window. This process is iterated over every sentence in the target text.

5 IMPLEMENTATION

5.1 LANGUAGE AND SOURCE CODE

5.1.1 Choice of language

The language modeler is programmed in Python; as programming languages go, it possesses the clearest syntax and is reasonably concise. Python runs in an interpreter and is slower than compiled languages such as C or Java, but this is remedied by the large amount of available libraries designed to circumvent this issue. For computationally demanding numerical problems such as are frequently found in machine learning, we can use libraries such as Numpy, Scipy, Theano, ... These offer implementations of frequently used numerical algorithms written in C, which are compiled during the execution of the program and cached.

In the past few years, Python has been switching from version 2 to version 3, which brought a lot of changes in syntax and generated a great deal of cross-compatibility problems. Despite this, many libraries are now available for Python 3, including the ones we are interested in using. Therefore, we chose to use Python 3.3 instead of Python 2.7.x; it offers superior Unicode and string processing capabilities to preceding versions.

5.1.2 Source code availability

The source code is available at `https://github.com/sinopeus/thrax`. The core numerical algorithms, encapsulated in Theano graphs, are based off Joseph Turian's implementation of an unsupervised model builder in Python 2. I rebuilt the entire program surrounding this numerical component to fit my needs: the result is a documented, cleaned up and overall improved program. I also added functionality for the supervised phase to complete the picture. The program's configuration system was enhanced and is now easily editable by hand as well as programmatically. With basic knowledge of programming, it is possible to train a network for any given NLP task by modifying the configuration.

5.2 THE FULL PROCESS

5.2.1 Preparing the training corpora

- conversion from TLG or TEI format to raw text
- conversion from Beta Code to Unicode
- stripping all characters which are not relevant, such as critical notation, paragraph markers etc.
- lowercase all words
- detailed tokenisation is not necessary
- convert annotation to a unified system
- create conversion script from classic annotation to one-hot vector annotation

5.2.2 Training the model

- window size of 11 words, due to the frequency of long-range dependencies in Greek
- represent features in 50-dimensional vectors (more dimensions could affect the computation time adversely)
- unsupervised iterations over increasing dictionaries: 5.000, 10.000, 20.000, 40.000, 80.000, 160.000, 360.000
- stop at the point of diminishing returns (computing the ranking for a dictionary of 360.000 given a corpus of size n requires 360.000 ranking formula calculations, which is bound to take a lot of time)
- continue training, now supervised but with the same hyperparameters and parameters as the supervised model

5.2.3 Preparing the object corpus

- conversion from TEI format to raw text
- one sentence per line!
- stripping all characters which are not relevant, such as critical notation, paragraph markers etc. but padding the text where necessary
- store tags sequentially
- basic tokenisation is handled during tagging
- run sentence through networks, then concatenate relevant outputs

- convert from one-hot vector notation to classic annotation
- iterate over every sentence
- done!

6 RESULTS

6.1 COMPUTATION TIME

6.2 RESULTING CORPUS

- corpus formatting: raw text/CoNLL

6.3 ACCURACY

6.3.1 Unsupervised model

- 'fuzzily' evaluate unsupervised model by taking nearest neighbours of a given set of words using the Euclidean metric
- use validation set to check scores

6.3.2 Supervised model

- evaluate supervised model by withholding a small set of sentences from the tagged training corpora
- this gives us a measure of the overall quality of the model

6.3.3 Goal corpus

- no previously available data: manual checking ...
- a set of about two to three hundred sentences sampled from all over the corpus
- shortcomings

7 ASSESSMENT

- hypothesis
- have we reached our goal?
- how thorough is this?
- contribution
- comparison

8 FURTHER WORK

This chapter is meant as a brief evocation of the potential of a fully annotated corpus of the papyri for further research.

8.1 COLLABORATIVE EDITING

The IDP project is heading into crowdsourcing territory at full steam, and excluding our own work from this movement would be inserting a shrill note into this symphony. All data is placed on GitHub at https://github.com/sinopeus/tjufy, freely accessible and editable for all.

This opens promising avenues of inquiry that do not have a direct relation to this thesis. For instance, it can in the future be integrated into SoSOL and absorbed into the larger codebase for the IDP project if it does not create too much overhead for the current developers (the technical back end of the IDP seems to be labyrinthine and adding new layers of complexity might be off-putting). It could even grow into a separate project which itself could be linked to papyri.info as the HGV, APIS and Trismegistos currently are, by a common system of indexation.

Making the code publically available to all also has the advantage of the public eye inspecting the texts; using solely automatic analysis is bound to deliver an inaccurate result, however small that inaccuracy may be, as creating a NLP engine perfectly capable of understanding language would be the equivalent of creating a perfect artificial intelligence. Therefore, considering the size of the corpus, one must rely upon the intelligence of the community. In the same way open source software is often among the best of software due to public inspection, the potential for a crowdsourced corpus is immense.

8.2 CORPUS-BASED GRAMMARS AND LEXICA

Expanding our method to other texts might bring the benefit of comprehensive corpus-based grammars and lexica, which can integrate available data on the fly and create a self-updating and reliable web of grammatical knowledge. Instead of focusing mainly upon a few choice authors or laboriously trudging through the huge wealth of ancient Greek literature to linearly create lexica and grammars, all of it could be harnessed at once in a quantitatively precise and easily visualisable way.

http://www.digitalhumanities.org/dhq/vol/003/1/000033/000033.html

8.3 HISTORICAL AND VARIATIONAL LINGUISTICS

The language of the papyri has an important role to play in the historical linguistics of Greek; once a full annotation has been achieved, it could be possible to implement the same methods used for synchronic language processing to map language changes in a statistical way; it could be possible to estimate the transition probabilities for diachronic grammatical evolutions, which has the potential to create a picture of the evolution of Greek that would be both comprehensive and precise. It even has potential on a comparative level; given the long history and meandering evolutionary trajectory of the Greek language, one could observe from the data catalysts for language evolution in one direction or the other and apply that comparatively.

One might also win valuable insight into language diversity in Egypt; using the paraliterary data already available from the Trismegistos, linguistic phenomena and evolutions could be visualised on a map and give insight into the diatopic, diastratic and diaphasic variation of Egyptian koinê, much in the way of modern dialect survey maps but directly linked to the original texts.

8.4 TEXTUAL CRITICISM

Textual criticism, too, could benefit from improved access to linguistic data; dubious *passus* could be disambiguated by comparing them to similar instances in papyri from the same period and adapting constructions and words from them. This technique is harnessed by Mimno and Wallach [2009], who use the techniques of statistical NLP solely for these specific critical problems. Though textual criticism will for the foreseeable future still necessitate trained papyrologists, the need for a very in-depth knowledge of the corpus of papyri can be greatly reduced by calling upon data from other parts of the corpus to present a series of statistically possible solutions for textual issues.

8.5 NAMED ENTITY RECOGNITION

Named entity recognition is a subdiscipline in natural language processing which is concerned with the automatic extraction and localisation of all kinds of names from texts. It has been used extensively in literary texts with a view to discern the importance of certain characters throughout the text. The KU Leuven's long-standing Prosopographia Ptolemaica project, which aims to be a repository of all inhabitants of

Egypt between 300 and 30 B.C., could easily benefit from these techniques. The abundant manual labour that has gone into the project could be fed as training data to and then supplemented by a named entity recognition engine that could also categorise personal names by any criteria and establish contextual relations between them. To take a very rudimentary example, the name 'Alexander' could be retrieved in all texts and a cluster of related names generated, so that related individuals may be placed in a web of relations; or one could ask, by combining the already present linguistic annotation, to display all adjectives which accompany the name 'Alexander'.

It could even go further than this and also include other particular names, such as places, distances, monetary units, weights, and so on. Historians could create a comprehensive overview of, for instance, the inflation of Egyptian currency, or map out trade connections using a search for all mentions of currency, weight and places which are in proximity to each other.

9 CONCLUSION

BIBLIOGRAPHY

Allan, R. J. and M. Buijs

2007 *The language of literature: linguistic approaches to classical texts*, Amsterdam studies in classical philology, 13, Brill.

Bakker, Egbert J.

1997 *Grammar as interpretation: Greek literature in its linguistic contexts*, Mnemosyne, bibliotheca classica Batava: Supplementum, 171, Brill.

Bakker, Stephanie J.

2009 *The noun phrase in ancient Greek: a functional analysis of the order and articulation of NP constituents in Herodotus*, Amsterdam studies in classical philology, 15, Brill.

Bakker, Stephanie J. and Gerry C. Wakker

2009 *Discourse cohesion in ancient Greek*, Amsterdam studies in classical philology, 19, Brill.

Bamman, David and Gregory Crane

2008 "Building a dynamic lexicon from a digital library", in *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, ACM, pp. 11–20. (Cited on p. 10.)

2009 "Computational Linguistics and Classical Lexicography", *Digital Humanities Quarterly*, 3, 1, http://www.digitalhumanities.org/dhq/vol/3/1/000033.html. (Cited on p. 10.)

2011 "The Ancient Greek and Latin Dependency Treebanks", in *Language Technology for Cultural Heritage*, ed. by Caroline Sporleder, Antal van den Bosch and Kalliopi Zervanou, Theory and Applications of Natural Language Processing, Springer Berlin Heidelberg, pp. 79–98, ISBN: 978-3-642-20227-8. (Cited on p. 10.)

Bamman, David, Francesco Mambrini and Gregory Crane

2009 "An Ownership Model of Annotation: The Ancient Greek Dependency Treebank", in *The Eighth International Workshop on Treebanks and Linguistic Theories (TLT 8)*. (Cited on p. 10.)

Bamman, David, Marco Passarotti and Gregory Crane

2008 "A Case Study in Treebank Collaboration and Comparison: Accusativus cum Infinitivo and Subordination in Latin", *The Prague Bulletin of Mathematical Linguistics*, 90, pp. 109–122. (Cited on p. 10.)

Bengio, Yoshua, Jérôme Louradour, Ronan Collobert and Jason Weston

2009 "Curriculum learning", in *Proceedings of the 26th annual international conference on machine learning*, ACM, pp. 41–48. (Cited on p. 19.)

Bengio, Yoshua, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin and Jean-Luc Gauvain

2006 "Neural probabilistic language models", in *Innovations in Machine Learning*, Springer, pp. 137–186.

Bod, Rens, Jennifer Hay and Stefanie Jannedy

2004 *Probabilistic Linguistics* (eds.), Cambridge, MA and London: MIT Press.

Boschetti, Federico

2009 *A Corpus-based Approach to Philological Issues*, PhD thesis, University of Trento.

Brunner, Ted

1993 "Ancilla to the Thesaurus Linguae Graecae", in *Accessing Antiquity*, ed. by Jon Solomon, Tucson: University of Arizona Press.

Chen, Yanqing, Bryan Perozzi, Rami Al-Rfou and Steven Skiena

2013 "The Expressive Power of Word Embeddings", *arXiv preprint arXiv:1301.3226*.

Collobert, Ronan

2011 "Deep learning for efficient discriminative parsing", in *International Conference on Artificial Intelligence and Statistics*, pp. 224–232.

Collobert, Ronan and Jason Weston

2008 "A unified architecture for natural language processing: deep neural networks with multitask learning", in *Proceedings of the 25th international conference on Machine learning*, ICML '08, ACM, Helsinki, Finland, pp. 160–167, ISBN: 978-1-60558-205-4, DOI: 10.1145/1390156.1390177, http://doi.acm.org/10.1145/1390156.1390177. (Cited on p. 13.)

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu and Pavel Kuksa

2011 "Natural Language Processing (Almost) from Scratch", *Journal of Machine Learning Research*, 12 [Aug. 2011], pp. 2493–2537, http://leon.bottou.org/papers/collobert-2011. (Cited on pp. 13, 14, 16.)

Crane, Gregory

1991 "Generating and parsing classical Greek", *Literary and Linguistic Computing*, 6, 4, pp. 243–245. (Cited on p. 9.)

Crönert, Wilhelm

1903 *Memoria Graeca Herculanensis*, Teubner: Leipzig. (Cited on p. 3.)

Deissmann, Adolf

1895 *Bibelstudien*, Marburg: N. G. Elwert. (Cited on p. 3.)

1897 *Neue Bibelstudien*, Marburg: N. G. Elwert. (Cited on p. 3.)

1929 *The N. T. in the Light of Modern Research*, London: Hodder & Stoughton. (Cited on p. 3.)

Denniston, John D. and Kenneth J. Dover

1950 *The Greek particles*, Gerald Duckworth.

Dieterich, K.

1898 *Untersuchungen zur Geschichte der griechischen Sprache von der hellenistischen Zeit bis zum 10. Jh. n. Chr.* Leipzig: Teubner. (Cited on p. 3.)

Dik, Helma

1995 *Word Order in Ancient Greek: A Pragmatic Account of Word Order Variation in Herodotus*, Amsterdam Studies in Classical Philology, 5, Amsterdam: J. C. Gieben.

Dik, Helma and Richard Whaling

2008 "Bootstrapping Classical Greek Morphology", in *Digital Humanities*. (Cited on p. 9.)

2009 "Implementing Greek Morphology", in *Digital Humanities*. (Cited on p. 9.)

Döttling, Christian

1920 *Die Flexionsformen lateinischer Nomina in den griechischen Papyri und Inschriften*, PhD thesis, Universität Basel.

Dover, Kenneth

1960 *Greek Word Order*, Cambridge University Press.

Dukes, Kais and Nizar Habash

2011 "One-Step Statistical Parsing of Hybrid Dependency-Constituency Syntactic Representations", in *Proceedings of the 12th International Conference on Parsing Technologies*.

Erhan, Dumitru, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent and Samy Bengio

2010 "Why Does Unsupervised Pre-training Help Deep Learning?", *J. Mach. Learn. Res.*, 11 [Mar. 2010], pp. 625–660, ISSN: 1532-4435, http://dl.acm.org/citation.cfm?id=1756006.1756025. (Cited on p. 19.)

Evans, Trevor V. and Dirk D. Obbink

2010 *The Language of the Papyri* (eds.), Oxford University Press. (Cited on pp. 1, 4.)

Fries, Charles C.

1940 *American English Grammar*, New York: Appleton Century Crofts.

Gignac, Francis Thomas

1964 *The Language of the post-Christian Greek Papyri: Phonology and Accidence*, PhD thesis, University of Oxford.

1970a "The Language of the Non-Literary Greek Papyri", in *Proceedings of the Twelfth International Congress of Papyrology*.

Gignac, Francis Thomas

1970b "The Pronunciation of Greek Stops in the Papyri.", *TAPA*, 101, pp. 185–202.

1974 "Loss of Nasal Consonants in the Language of the Papyri.", in *Akten des XIII. Internationalen Papyrologenkongresses Marburg/Lahn 1971*.

1976 *A Grammar of the Greek Papyri of the Roman and Byzantine Periods. I. Phonology.* Milano: Goliardica. (Cited on p. 3.)

1981a *A Grammar of the Greek Papyri of the Roman and Byzantine Periods. II. Morphology.* Milano: Goliardica. (Cited on p. 3.)

1981b "Some Interesting Morphological Phenomena in the Language of the Papyri.", in *Proceedings of the Sixteenth International Congress of Papyrology*.

1985a "The Papyri and the Greek Language.", *YCS*, 28, pp. 155–165.

1985b "The Transformation of the Second Aorist in Koine Greek.", *BASP*, 22, 49-54.

1986 "Morphological Phenomena in the Greek Papyri Significant for the Text and Language of the New Testament.", in 48, pp. 499–511.

1987 "Analogical Levelling in "-mi" Verbs", in *Miscel.lània Papirològica Ramon Roca-Puig*, Barcelona: Fundacio Salvador Vives Casajuana, pp. 133–140.

Harsing, C.

1910 *De optativi in chartis Aegyptiis usu*, PhD thesis, Universität Bonn.

Haug, Dag Trygve Truslew *et al.*

PROIEL: Pragmatic Resources in Old Indo-European Languages, http://foni.uio.no:3000/, info at http://www.hf.uio.no/ifikk/proiel/. (Cited on p. 8.)

Haykin, Simon S.

1999 *Neural networks: a comprehensive foundation*, 2nd ed., Prentice Hall, ISBN: 9780132733502, http://books.google.be/books?id=bX4pAQAAMAAJ.

Hinton, Geoffrey E

2007 "Learning multiple layers of representation", *Trends in cognitive sciences*, 11, 10, pp. 428–434. (Cited on p. 14.)

Hopcroft, John E., Rajeev Motwani and Jeffrey D. Ullman

2001 *Introduction to automata theory, languages, and computation*, 2nd ed., Boston: Addison-Wesley.

Horn, R. C.

1926 *The Use of the Subjunctive and Optative Moods in the Non-Literary Papyri*, PhD thesis, University of Pennsylvania.

Huang, Eric H, Richard Socher, Christopher D Manning and Andrew Y Ng
2012 "Improving word representations via global context and multiple word prototypes", in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, Association for Computational Linguistics, pp. 873–882.

Humbert, Jean
1930 *La disparition du datif en grec*, Collection linguistique publiée par La Société de linguistique de Paris, 33, Paris: Champion.

Integrating Digital Papyrology
2008 *Background and Funding*, http://idp.atlantides.org/trac/idp/wiki/BackgroundAndFunding.

Jurafsky, Daniel and James H. Martin
2009 *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd ed., Pearson.

Kapsomenos, Stylianos G.
1938 *Voruntersuchungen zu einer Grammatik der Papyri der nachchristlichen Zeit*, Münchener Beiträge zur Papyrusforschung und antiken Rechtsgeschichte, München: C. H. Beck. (Cited on p. 4.)
1957 "Ἔρευναι εἰς τὴν γλῶσσαν τῶν ἑλληνικῶν παπύρων. Σείρα Πρώτη", *EEThess*, vii, pp. 225–372. (Cited on p. 4.)

Koshy, Thomas
2004 *Discrete Mathematics with Applications*, Amsterdam, Boston: Elsevier Academic Press.

Kuhring, Walter
1906 *De praepositionum Graecarum in chartis Aegyptiacis usu, quaestiones selectae*, PhD thesis, Universität Bonn.

Lee, John
2008 "A nearest-neighbor approach to the automatic analysis of ancient Greek morphology", in *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, Association for Computational Linguistics, pp. 127–134. (Cited on p. 9.)

Lee, John and Dag Haug
2010 "Porting an Ancient Greek and Latin Treebank.", in *LREC*, ed. by Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner and Daniel Tapias, European Language Resources Association, ISBN: 2-9517408-6-7, http://dblp.uni-trier.de/db/conf/lrec/lrec2010.html#LeeH10. (Cited on p. 10.)

Ljungvik, Herman
1929 "Ur papyrusbrevens språk", *Eranos*, xxvii, pp. 166–181.

Ljungvik, Herman

1932 *Beiträge zur Syntax der spätgriechischen Volkssprache.* Vol. 27, Skrifter utgivna av K. Humanistiska Vetenskaps-Samfundet i Uppsala, 3, Uppsala & Leipzig: Humanistiska Vetenskaps-Samfundet. (Cited on p. 4.)

Mahoney, Anne

2000 *Generalizing the Perseus XML Document Manager*, http://ldc.upenn.edu/exploration/expl2000/papers/mahoney/mahoney.htm.

Mandilaras, Basileios G.

1961 "Ἔρευναι εἰς τὴν γλῶσσαν τῶν μὴ φιλολογικῶν παπύρων. Τελικοὶ σύνδεσμοι μετ᾽ ἀπαρεμφάτου", *Athena*, 65, pp. 169–176, 169–176.

1967 "Παρατηρήσεις εἰς τοὺς Ἑλληνικοὺς παπύρους", *Athena*, 69, pp. 94–116.

1969-70 "Καινὴ Διαθήκη καὶ παπύροι. Συγκριτικὴ ἐξέτασις γλωσσικῶν τινῶν φαινομένων", *Athena*, 71, pp. 130–164.

1971-1972 "Ἐπὶ τῆς φωνητικῆς τῶν παπύρων", *EEAth*, 22, pp. 256–266.

1972 *Studies in the Greek Language.* Ἀθήνα: Ξενόπουλος.

1973 *The verb in the Greek non-literary papyri*, Athens: Hellenic Ministry of Culture and Sciences. (Cited on pp. 3, 4.)

1974 "Confusion of Aorist and Perfect in the Language of the Non-Literary Greek Papyri.", in *Akten des XIII. Internationalen Papyrologenkongresses Marburg/Lahn 1971*, pp. 251–261.

Manning, Christopher D. and Hinrich Schütze

1999 *Foundations of Statistical Natural Language Processing*, Cambridge, MA and London: MIT Press.

Manolessou, Io and Notis Toufexis

2011 "Corpus linguistics in historical dialectology: a case study of Cypriot", in *Studies in Modern Greek Dialects and Linguistic Theory*, Nicosia: Research Centre of Kykkos Monastery.

Mayser, Edwin

1938 *Grammatik der griechischen Papyri aus der Ptolemäerzeit, mit Einschluss der gleichzeitigen Ostraka und der in Ägypten verfassten Inschriften.* Berlin, Leipzig: De Gruyter. (Cited on p. 3.)

McCarthy, Michael and Anne O'Keeffe

2010 "What are corpora and how have they evolved?", in *The Routledge Handbook of Corpus Linguistics*, ed. by Anne O'Keeffe and Michael McCarthy, London and New York: Routledge. (Cited on pp. 4, 5.)

Mimno, David and Hanna Wallach

2009 "Computational Papyrology (presentation)", in *Media in Transition 6: Stone and Papyrus, Storage and Transmission*, Cambridge, MA. (Cited on p. 36.)

Mnih, Andriy and Geoffrey Hinton

2007 "Three new graphical models for statistical language modelling", in *Proceedings of the 24th international conference on Machine learning*, ACM, pp. 641–648.

Moulton, J. H.

1901 "Grammatical Notes from Papyri", *C. R.*, xv, pp. 31–8, 434–442.

1903 "Notes from the Papyri", *The Expositor*, vii, pp. 104–121.

1904 "Grammatical Notes from Papyri", *C. R.*, xviii, pp. 106–122, 151–155.

Nakov, Preslav, Ariel Schwartz and Brian Wolf

2005 "Supporting Annotation Layers for Natural Language Processing", in *Annual Meeting of the Association of Computational Linguistics*.

Natural Language Toolkit

2012 *Natural Language Toolkit*, http://www.nltk.org/.

Packard, David W

1973 "Computer-assisted morphological analysis of ancient Greek", in *Proceedings of the 5th conference on Computational linguistics-Volume 2*, Association for Computational Linguistics, pp. 343–355. (Cited on p. 9.)

Palmer, Leonard Robert

1934 "Prolegomena to a Grammar of the post-Ptolemaic Papyri", *J. Th. S.*, xxxv, pp. 170–5. (Cited on p. 4.)

1945 *A Grammar of the post-Ptolemaic Papyri: Accidence and Word-Formation*, London: Oxford University Press. (Cited on p. 4.)

Porter, S. E. and M. B. O'Donnell

2010 "Building and Examining Linguistic Phenomena in a Corpus of Representative Papyri", in *The Language of the Papyri*, Oxford University Press, pp. 287–311. (Cited on p. 4.)

Rijksbaron, A.

2007 *The syntax and semantics of the verb in classical Greek: an introduction*, University of Chicago Press.

Rossberg, C.

1909 *De praepositionum Graecarum in chartis Aegyptiacis Ptolemaeorum aetatis usu*, PhD thesis, Universität Jena.

Russell, S.J. and P. Norvig

2010 *Artificial Intelligence: A Modern Approach*, 3rd ed., Prentice Hall Series in Artificial Intelligence, Pearson Education/Prentice Hall, ISBN: 9780136042594, http://books.google.be/books?id=8jZBksh-bUMC.

Salonius, A. H.

1927 *Zur Sprache der griechischen Papyrusbriefe*, vol. i, Die Quellen, Akademische Buchhandlung. (Cited on p. 3.)

Schmid, Helmut

1994 "Probabilistic Part-of-Speech Tagging Using Decision Trees", in *Proceedings of International Conference on New Methods in Language Processing*. (Cited on p. 9.)

1995 "Improvements in Part-of-Speech Tagging with an Application to German", in *Proceedings of the ACL SIGDAT-Workshop*. (Cited on p. 9.)

Schubart, W.

1918 "Einführung in die Papyruskunde", in Berlin: Weidmann, chap. "Die Sprache der Papyri, pp. 184-226).

Serz, H.

1920 *Der Infinitiv in der griechischen Papyri der Kaiserzeit (bis Diokletian)*, PhD thesis, Universität Erlangen.

Sicking, C.M.J. and J. M. Ophuijsen

1993 *Two studies in Attic particle usage: Lysias & Plato*, Mnemosyne, bibliotheca classica Batava: Supplementum, 129, E.J. Brill.

Stanford NLP Group

2011 *Statistical natural language processing and corpus-based computational linguistics: An annotated list of resources*, `http://www-nlp.stanford.edu/links/statnlp.html`.

Thumb, Albert

1901 *Die griechische Sprache im Zeitalter des Hellenismus: Beiträge zur Geschichte und Beurteilung der KOINH.* Strassburg: Trübner. (Cited on p. 3.)

1906 "Principienfragen der Κοινή Forschung.", *Neue Jahrbücher für das klassische Altertum*, 17, pp. 246–63. (Cited on p. 3.)

Toufexis, Notis

2010 "One Era's Nonsense, Another's Norm: Diachronic study of Greek and the Computer", in *Digital Research in the Study of Classical Antiquity*, ed. by Gabriel Bodard and Simon Mahony, London: Ashgate.

Trismegistos

About Trismegistos, http://www.trismegistos.org/about.php.

Turian, Joseph, Lev Ratinov and Yoshua Bengio

2010 "Word representations: a simple and general method for semi-supervised learning", in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, pp. 384–394. (Cited on p. 14.)

Turing, Alan Mathison

1950 "Computing Machinery and Intelligence", *Mind*, 59, 236 [Oct. 1950], pp. 433–460. (Cited on p. 7.)

Turner, Eric Gardner

1980 *Greek Papyri: An Introduction*, 2nd ed., Oxford University Press.

Van Hal, Toon

2010 "A propos des travaux linguistiques sur corpus en grec ancien, en latin et en néo-latin", in *Actes de la première journée d'étude franco-allemande de linguistique*.

Völker, Franz

1900 *Papyrorum graecarum syntaxis specimen*, PhD thesis, Universität Bonn.

1903 *Syntax der griechischen Papyri*, vol. i, Der Artikel, Münster: Westfälischen Vereinsdruckerei.

W3C

2008 *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, http://www.w3.org/TR/xml/.

Wahlgren, Staffan

2010 "Byzantine Literature and the Classical Past", in *The Blackwell Companion to the Ancient Greek Language*, ed. by Egbert J. Bakker, Blackwell, pp. 527–39.

Witkowski, Stanislaus

1902 *Prodromus grammaticae papyrorum graecarum aetatis Lagidarum*, Vienna: Universitäts-Buchdruckerei.

A CONCEPTS AND TECHNIQUES

A.1 MATHEMATICS

A.1.1 Set theory

Consider an object o and a set A. We write 'object o is an element of set A' as $o \in A$. Sets themselves are also objects and can belong to other sets. Consider two sets, A and B. We write that A is a subset B as $A \subseteq B$; this implies that B contains all elements in A and does not exclude the possibility of $A = B$; if A is a proper subset of B, i.e. all elements of A are in B but not all elements of B are in A, we write $A \subset B$. Mirroring these symbols from right to left gives us the symbols for supersets and strict supersets, respectively. The empty set, which contains no elements, is written as $\varnothing$.

There exist different binary operators on sets (i.e. operators on two sets) which return another set. The most common operators are:

- the union of sets, written as $A \cap B$, denotes a set which contains all elements which are in A and B;
- the intersection of sets, written as $A \cup B$, denotes a set which contains all elements which are in both A and B;
- the difference of sets, written as $A \setminus B$, denotes a set which contains every element from A excluding those which are also in B.
- the Cartesian product of sets, written as $A \times B$, is the set containing all ordered pairs of elements from A and B.

A.1.2 Probability

A **probability** is a measure for the likelihood of an event for an experiment, which we intuitively understand to be an action whose outcome we want to observe. Such events are then elements of a set containing all possible outcomes of an experiment; an event can be a point or subset of that set. We call this set the **sample space**, denoted S. We denote the probability of an event E as P(E).

Axiomatically, we can define probability as follows:

1. For every event E, $P(E) \geqslant 0$; no event can have a negative probability.
2. $P(S) = 1$; that is to say, every experiment has an event.

3. For any sequence of disjoint events A_i (that is to say, there is no overlap between events), the probability of any one of these events occurring is the sum of their respective probabilities.

A few other important properties of probabilities and events are the following:

1. The complement of an event A, which is the union of all elements of S which are not an element of A, is denoted A^C; $P(A^C$ is ther probability of this event occurring and is equal to $1 - P(A)$.

2. For any event E, $0 \leqslant P(E) \leqslant 1$.

3. Given any two events A and B, $A \subset B \rightarrow P(A) \leqslant P(B)$.

Given probabilities of a number of events, we can establish relationships between these probabilities and compute related probabilities using the rules certain rules. A classic rule is the **multiplication rule**, which states that if we perform an experiment in k parts and the i^{th} part of the experiment has n_i possible outcomes, and the outcomes of prior parts of the experiment do not affect latter ones, the probability of any specific sequence of partial outcome will be the product of all outcome counts n_i with i ranging from 1 to k.

Set theory is important when we want to know the probability of an event E which can be constructed from a set of sets A_i using set operators. The third axiom of probability has already given us the solution for disjoint events; events may also overlap, and in this case, we need a more sophisticated formula. For the union of any n events A_i, the following holds:

$$P(\bigcup_{i=1}^{n} A_i) = \sum_{i=1}^{n} P(A_i) \tag{7}$$

For the intersection of these n events A_i, it can be proven that:

$$P(\bigcap_{i=1}^{n} A_i) = \lim_{n \to \infty} P(A_i) \tag{8}$$

Knowing both these rules is important when considering **conditional probability**; for two events A and B, suppose we know B has occurred and we want to know what the probability of A occurring is given this information. We call this the conditional probability of A given B and write it $P(A|B)$. If $P(B) > 0$, then we define it as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{9}$$

Using this formula, we can directly derive **Bayes' theorem** as follows:

$$\begin{aligned} P(A|B) &= \frac{P(A \cap B)}{P(B)} \\ P(A|B)P(B) &= P(A \cap B) \\ P(A|B)P(B) &= P(B \cap A) \\ P(A|B)P(B) &= P(B|A)P(A) \\ P(B|A) &= \frac{P(A|B)P(B)}{P(A)} \end{aligned} \tag{10}$$

A.1.3 Calculus and linear algebra

- Derivatives and computing extrema
- Jacobian matrices
- Numerical methods
- Vector spaces

A.1.4 Statistics

Regression

Regression is a classic technique from statistics, often visualised as 'fitting a line to a set of points'. Classification is a related technique which uses regression to classify new data points. This is essentially what any probabilistic model for natural language processing does, in one form or another. What follows is an overview of various types of regression and corresponding methods for classification.

We start with **univariate linear regression**. Given a set of n points in the plane, we want to find a hypothesis that best corresponds to the location of these points, and we want this hypothesis to be a linear function. This function is then of the form:

$$h_w(x) = w_1 x + w_0 \tag{11}$$

Unless all points are collinear, it is of course impossible to find a function of this form that gives a correct mapping for each point. The best we can do is find the values of w_0 and w_1 for which the empirical loss on the mappings is minimal. The traditional way of doing this is to define a function that computes the squares of the errors and sums it over all data points; this is called an L_2 **loss function**. We now want to find the values of w_0 and w_1 for which this function attains a minimum. We can find these minimal points by solving for the roots of the partial derivative functions of this loss function with respect to w_0 and w_1. This problem is mathematically relatively simple and has

a unique solution. This solution is valid for all loss functions of this type.

Problems arise when we are trying to create a nonlinear model. In this case, the minimum loss equations frequently do not have a unique solution. We can of course still model the problem algebraically, and the goal is the same: finding the roots of the partial derivative function. Now, however, we need to use a more sophisticated method: **gradient descent**. We can visualise this technique as 'descending a hill'; the 'hill' is the graphical representation of the root of the system of partial derivatives, and by 'descending' this hill, i.e. by iteratively picking values which bring us closer to the bottom part of the valley next to the hill, which corresponds to the minimal point of the function, eventually convergence will be reached on the minimum and we will found the correct weights for our function. The difference by which we change the value at each iteration is called the **step** or **learning rate** and determines how fast we will converge; it may be either a fixed constant or a mutable value which can increase or decay according to the current state of our descent.

Multivariate linear regression poses a similar problem; only this time the function is not dependent on a single variable, but on two or more. Such a function is a bit more complex, but we can find a solution to the regression problem using analogous techniques. Suppose the function has n variables. Each example x_j must be a vector with n values. At this point, we are looking at a function of the following form:

$$h_w(x_j) = w_0 + w_1 x_{j,1} + w_2 x_{j,2} + \ldots + w_n x_{j,n} = w_0 + \sum_i w_i x_{j,i} \quad (12)$$

We want to simplify this to make algebraic manipulations easier. We therefore prepend an extra component $x_{j,0} = 1$ to the vector x_j; now using vector notation we can simplify the previous equation to:

$$h_w(x_j) = \sum_i w_i x_{j,i} = w \cdot x_j \quad (13)$$

What we are now looking for is a vector w containg the weights of our function which minimises the empirical loss, as in univariate linear regression. We can equivalently use gradient descent; only now, of course, the computational cost of that technique will be higher. A common problem can now appear: **overfitting**, that is, giving an irrelevant dimension of the vector w too much weight due to chance errors in the computation. This can be compensated by taking into account the complexity of the hypothesis; a statistical equivalent to Ockham's razor, if you will.

Classification

We can define an analogous process for classification; only now the function must not fit to the data itself but must create a **decision boundary** between data points. If there exists a linear function which

satisfies this property for a given data set , we call the bounding line or surface generated by this function a **linear separator**, and the data set **linearly separable**. The hypothesis function is now of the form:

$$h_w(x) = 1 \text{ if } w \cdot x \geqslant 0, 0 \text{ otherwise.} \tag{14}$$

We can view this as a function $\texttt{threshold}(w \cdot x)$ which is equal to 1 only if $w \cdot x = 0$ Note that while the separating function is linear, the hypothesis function is not, and in fact has the distinctly unappealing property of not being differentiable. We can therefore not apply the technique of gradient descent here. Furthermore, this type of function has exactly two outputs: 1 or 0. For our purposes, we need subtler methods of classification. This type of hypothesis function is therefore not fit for our purposes, but it does give a good idea of what classification is.

The best option is replacing the hard threshold function with the sigmoid or logistic function, which offers a good approximation and is differentiable at every point. This function is of the form:

$$g(x) = \frac{1}{1 + e^{-x}} \tag{15}$$

Such that our new hypothesis function is:

$$h_w(x) = g(w \cdot x) = \frac{1}{1 + e^{-w \cdot x}} \tag{16}$$

If we use this function, we are performing **linear classification with logistic regression**.

Logistic regression and the chain rule

Clustering

A.1.5 Formal language theory

- Languages and strings
- Regular languages
- Context-free grammar and languages
- The Chomsky hierarchy

A.2 NATURAL LANGUAGE PROCESSING

A.2.1 N-grams

A.2.2 Hidden Markov Models

A.2.3 Viterbi decoding

A.3 ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

A.3.1 What is machine learning?

- Supervised learning
- Unsupervised learning

A.3.2 Neural networks

An artificial neural network is massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use.

For the design of an architecture which allows us to solve the problems above, we have taken our cues largely from recent work in machine learning as applied to natural language processing. In particular, we follow the approach set out in Weston & Collobert 2008 and expanded in Weston et al. 2011, that is, the use of deep neural networks for joint training on our chosen corpus. This section is dedicated to a more expository overview of that architecture for the mathematical layman.

An artificial neural network is a massively parallel processing system constructed from simple interconnected processing units (called neurons) which has the ability to learn by experience and store this knowledge for later use. The term 'neural network' is due to the resemblance of this architecture to the most powerful biological processor known to exist, the human brain, which has a way of functioning which is broadly analogous to this process.

Artificial neural networks find their origin in a mathematical model dating from before the first wave of artificial intelligence in 1956, the McCulloch-Pitts Threshold Logical Unit, known also as the McCulloch-Pitts neuron. Warren McCulloch was a psychiatrist and neuroanatomist; Walter Pitts a mathematical prodigy. Both met at the University of Chicago, where a neural modeling community led by the mathematical physicist N. Rashevsky had been active in the years preceding the publication in 1943 of the seminal paper A logical calculus of the ideas immanent in nervous activity.

Formally, we can define a neuron as a triplet (v,g,w) where:

- v is an input function which takes a number of inputs and computes their sum;

- g is an activation function which is applied to the output of the input function and determines if the neuron 'fires';
- w is an output function which receives its input from the activation function and distributes it over a number of outputs.

Given its structure, we can also see a neuron as a composite function $F = w \circ g \circ v$. The combination of several of these units using directed links forms a neural network. A link connecting a node i to a node j transfers the output a_i of node i to node j scaled by a numeric weight $w_{i,j}$ associated with that specific link. This is the general model of a neural network; countless variations on this theme have been developed for different purposes, mainly by modifying the activation function and the interconnection of neurons.

The activation function g typically will be a hard threshold function (an example of this is the original McCulloch-Pitts neuron), which makes the neuron a perceptron, or a logistic (also known as a sigmoid) function, in which case we term the neuron a sigmoid perceptron. Both these functions are nonlinear; since each neuron itself represents a composition of functions, the neuron itself is a non-linear function; and since the entire network can also be seen as a composite function (since it takes an input and gives an output) the network can be viewed as a nonlinear function. Additionally, choosing to use a logistic function as an activation function offers mathematical possibilities, since it is differentiable. This offers similar possibilities as the use of the logistic function for regression (cf. *supra*).

The links between nodes can be configured in different ways, which each afford distinct advantages and disadvantages. Broadly, we can distinguish two models. The simplest model is the **feed-forward network**, which can be represented as an acyclic directed graph. The propagation of an input through this kind of network can be seen as a stream, with posterior (downstream) nodes accepting outputs from prior (upstream) nodes. This type of network is the most widely-used and is used in the architecture. A more complex type is the **recurrent network**, which feeds its output back to its input and thus contains a directed cycle; this type of network has interesting applications (for example in handwriting recognition), as they resemble the neural architecture of the brain more closely than feedforward networks do.

Feed-forward networks are often organised (to continue the stream analogy) in a kind of waterfall structure using layers. The input is the initial stream, the output is the final stream; in between, we may place hidden layers, which are composed of neurons which take inputs and outputs as any neuron does, but whose output is then immediately transferred to a different neuron. Throughout the network, we can equip the neurons in each layer with distinct activation functions and link weights and in this way mold the learning process of the network to our purpose.

Single-layer networks contain no hidden layers; the input is directly connected to the output. Therefore, the output is a linear combination of linear functions. This is undesirable in many cases. The main prob-

lem, demonstrated early on in the development of neural network theory, is the fact that such a network is unable to learn functions that are not linearly separable; one such function is the XOR function, which is a very simple logical operator. Despite this, such neural networks are useful for many tasks, as they offer an efficient way of performing logistic regression and linear classification.

Our interest lies in multi-layer networks, however. Multi-layer networks contain one or more layer between the input and output layer, which are called hidden layers. By cascading the input through all these layers, we are in fact modeling a nonlinear function which consists of nested nonlinear soft threshold functions as used in logistic regression. The network can now be used to perform **nonlinear regression**. Different algorithms exist which can be used to train the network; the most important one is the **backpropagation algorithm**, which is the equivalent of the loss reduction techniques used in linear regression.

Suppose that a neural network models a vector-valued hypothesis function h_w which we want to fit to an example output vector y. We can create a L_2 loss function E by taking the error on this vector and squaring it. This function can be quite complex, but by taking partial derivatives of this function, we can consider the empirical loss on each output separately, like so:

$$\begin{aligned}\frac{\partial}{\partial w}E(w) &= \frac{\partial}{\partial w}|y - h_w(x)|^2 \\ &= \frac{\partial}{\partial w}\sum(y_k - a_k(x))^2 \\ &= \sum\frac{\partial}{\partial w}(y_k - a_k(x))^2\end{aligned} \tag{17}$$

If the output function is to have m outputs, instead of handling one large problem, we can subdivide the problem into m smaller problems. This approach works if the network has no hidden layers, but due to the fact that nothing is really known about the hidden layers if we only look at the output layer, a new approach is necessary. This is called backpropagation, a shorthand term for backward error propagation.

- backpropagation

A.3.3 Deep learning

B NEURAL NETWORK LAYERS