COMBING TEXTS: A QUEST TO INCREASE THE TIMELINESS AND ACCURACY OF GEOTAGGING MULTILINGUAL TOPONYMS AND TAGGING PERSONS IN LARGE CORPORA USING TENSORS FOR DISAMBIGUATION

A Dissertation Submitted
to the Graduate School
University of Arkansas at Little Rock

in partial fulfillment of requirements
for the degree of

DOCTOR OF PHILOSOPHY
in Computer and Information Science

in the Department of Computer Science
of the College of Engineering and Information Technology

November 2019

Anthony D. Davis

B.A. Computer Science and History, Lyon College, 2004
M.T.S., Digital Humanities and Historical Studies, Vanderbilt University, 2014

This dissertation, "COMBING TEXTS: A QUEST TO INCREASE THE TIMELINESS AND ACCURACY OF GEOTAGGING MULTILINGUAL TOPONYMS AND TAGGING PERSONS IN LARGE CORPORA USING TENSORS FOR DISAMBIGUATION," by Anthony D. Davis, is approved by:

Dissertation Advisor:	Dr. Xiaowei Xu Professor of Information Science
Dissertation Committee	Dr. John Talburt Professor of Information Quality
	Dr. Nitin Agarwal Jerry L. Maulden-Entergy Chair Professor of Information Science
	Dr. Elizabeth Pierce Associate Professor, Chair of Information Science
	Dr. Clifford Anderson Professor of Religious Studies Adjunct Professor of Computer Science Vanderbilt University
Program Coordinator:	Dr. John Talburt Professor of Information Science
Graduate Dean:	Dr. Lawrence Whitman Graduate Dean of Engineering and Information Technology

COMBING TEXTS: A QUEST TO INCREASE THE TIMELINESS AND ACCURACY OF GEOTAGGING MULTILINGUAL TOPONYMS AND TAGGING PERSONS IN LARGE CORPORA USING TENSORS FOR DISAMBIGUATION by Anthony D. Davis, August 2019

ABSTRACT

This research demonstrates an effective algorithm and provides a tool for efficient and accurate disambiguation of domain specific texts. The goals outlined in the research are: 1) to demonstrate that a simple knowledge base can be used to tag texts; 2) to demonstrate that a tensor can be used to efficiently and accurately tag entities in raw text documents, and; 3) create and provide a machine learning digital humanities tool for scholars interested in early Christian documents in English and Greek from late-antiquity. The research shows that a simple knowledgebase using a comma delimited file along with a rank-3 tensor using a Bayesian method for probability can result in a high level of recall and precision on domain specific projects. While this specific tool is for early Christian writings (English translations and Greek) from late antiquity, it can easily be modified to fit any corpora of texts. The end user need only create a simple comma delimited file with entity names, surnames, aliases, and connections to tag the texts in TEI standard. There are numerous named entity resolution/disambiguation tools available to scholars. However, each of them are suitable for "big data" and do not provide the accuracy needed in domain specific projects; especially projects dealing with non-contemporary entities. Therefore, this research demonstrates a novel approach and a novel tool for tagging raw text in large corpora.

DEDICATION

This work is dedicated to my wife, Hillary, and to my amazing children: Laurel, Thomas, and Clara.

ACKNOWLEDGMENTS

There are numerous people I would like to acknowledge as being integral to the success of this research and dissertation. First and foremost my family; none of this would be possible without a considerable amount of time away from Hillary, Laurel, Thomas, and Clara to work on this research. Their understanding and support were crucial. On that same line, this would not have been possible without the support of my mother-in-law, Barbi Vinsant, keeping the kids on my days off; I cannot thank her enough. I would like to recognize the Arkansas Geographic Information System Office, the Arkansas Game and Fish Commission, and to Lyon College for the flexibility and understanding they offered while pursuing my doctorate and working full-time. My college mentor and now colleague at Lyon College, Dave Sonnier, for making sure that I devote as much time as necessary to the work, among many other things throughout the years. To my students at Lyon College, especially Zach Ward, for taking an interest in my research and even providing ideas for future avenues of research. Their interest in the subject was a major factor and inspiration to complete the project.

My dissertation adviser, Dr. Xiaowei Xu, has been an exceptional and understanding advisor. He has been so gracious and patient with me knowing that I had a full-time job, a growing family, etc. This research would not have been possible without his guidance and mentorship these past few year.

Finally, I would like to acknowledge my mentor and my masters advisor at Vanderbilt University, Dr. David Michelson. His classes, his research, his mentorship, and his friendship are directly responsible for not only this dissertation topic, but for the entire trajectory of my professional and academic path. Without our paths crossing, life would look very different.

TABLE OF CONTENTS

LIST OF FIGURES OR ILLUSTRATIONS

LIST OF TABLES

PREFACE

INTRODUCTION

The field of digital humanities -- or humanities computing -- combines typical humanities research with technology to analyze texts, maps, polling data, language, and a variety of other avenues. Many in the humanities are demonstrating that adding such analysis can enhance their research and is a tool for humanists use, not a replacement for typical humanities research. As the number of people majoring in the humanities continues to decline [1], adding the digital component to research may open avenues to students. Digital humanities does allow for collaborative avenues for teachers and students to collaborate on research projects. For instance, a Syriacist -- one who studies Syriac manuscripts -- has a large corpora of texts at their disposal and may want to narrow the sources down by region or by a specific person. While many of these sources are digitized, few of them are processed for geographic or person tagging. The process to manually scan all of the sources to determine which fall in their sphere of interest could take years. One also finds that this is a huge undertaking and prone to human errors. Just tagging geographic toponyms would be of great value to researchers. According to Zaila et al, “It is estimated that more than 70% of all information in the world has some kind of geographic features” [2]. The goal of this research is: 1) to demonstrate that one can use a simple knowledge base can be used to tag texts; 2) to demonstrate that one can use a tensor can to efficiently and accurately tag entities in raw text documents, and; 3) create and provide a machine learning digital humanities tool for scholars interested in early Christian documents in English and Greek from late-antiquity. Access to this domain based -- religious studies/history -- tool will allow scholars to read any raw text from the field, create automatically generate TEI documents of these texts, and upload them into no-SQL databases, such as eXist-DB, for further analysis.

Given that the goal is to analyze whether a simple knowledge base in conjunction with a tensor can learn over each each text, this research will perform tests on multiple texts using the tensor functionality that learns as it grows versus a static knowledge base that only uses supervised input. Using this tool, one can demonstrate that a domain specific knowledgebase using a tensor based disambiguation process will outperform the state-of-the-art named entity recognition tools for large corpora domains.

BODY OF TEXT

Current State of Topic

The fields of machine learning and artificial intelligence are growing at a rapid pace. As one can imagine, the current field of Named Entity Resolution (NER) and Disambiguation (NED) is seeing a flurry of activity, as well. The current state of geoparsers, which is the process of using NER/NED for geographic entities, seems to rely almost exclusively in matching place names with other place names in the knowledge base. From there, there are a variety of algorithms used, as described below in the “Geotagging” subsection.

The broader topic of NER/NED is reviewed in the “Named Entity Resolution/Disambiguation” subsection below. The research here seems to rely on pre-existing datasets for the knowledge base, such as wikipedia or DBpedia, Freebase, etc [3, 4].

Geotagging

A recent journal article published in 2015 by Alex, et al titled, *Adapting the Edinburgh Geoparser for Historical Georeferencing* is salient to the proposed research objective. The article describes a long-standing geocoder that was created and optimized for modern day texts, specifically news articles. However, the purpose of the article is to demonstrate the geocoder’s use in other types of texts, such as historical corpora. The article demonstrates that geocoders have two steps: the “recognition step” and the “disambiguation step, generally referred to as georesolution”. [5]. The geocoder uses XML -- which the current proposal wishes to use, as well -- in the recognition step. However, there is a large difference in the second phase. The article is an example of the hierarchical approach and discusses how “the georesolver uses heuristics combined with weighting of information to arrive at its rankings. For example, populated places are preferred over places described in a gazetteer as facilities, and larger places (by population) are weighted more highly than smaller ones" [5]. To further enhance the geocoder's disambiguation, the geocoder takes all of the places in the document "so that all locations mutually constrain one another to be as close together as possible" later referred to in the article as "location clustering" [5]. Therefore, the geocoder uses the type/size of location and proximity to other locations on the globe to disambiguate places. It is important to note that the project does uses persons in the disambiguation phase, but only to discount them from being geocoded. The article states "at the recognition step, for example, we found that identifying person names and location names in parallel, even though there is no need to extract person name information for the intended application, helped to improve the overall quality of the text mined output" [5]. So, while identifying persons in the process helped disambiguate a person from a place, the authors did not use persons to disambiguate one place from another. The article does use a gazetteer proposed in the future research section, Pleiades, for testing on ancient places. To determine the accuracy of the process, the team "used the output of an earlier project, Hestia, to gauge accuracy over comparable ancient text. The Hestia project used a hand-annotated version of Herodotus' Histories from the Perseus Digital Library" [5]. According to the article, "The precision and recall scores for place name recognition over this text were 77.74% and 95.58% respectively, giving F-score of 85.74%" [5]. This project and score is used -- in conjunction with subsequent scores presented in the "Current State of Research" section -- as a baseline for comparing this proposal's tensor methods against the Edinburgh Geoparser, et al.

In Melo, et al's *Automated Geocoding of Textual Documents: A Survey of Current Approaches* published in 2016, Melo, et al look at the current environment surrounding geocoding texts. They lay out four approaches to geocoding texts: 1) "...heuristics over place names mentioned in the text (e.g. names of cities and states), (2) probabilistic language modeling approaches…(3) combinations of different models and heuristics, including clustering procedures, feature selection approaches, and/or language models built from different sources, and (4) recent approaches based on discriminative classification models" [6]. In describing some of the early work in the field -- early 2000's -- Melo, et al demonstrate that one method involved three steps: "1) place name spotting; (2) place name disambiguation; and (3) geographical focus determination" [6]. In the disambiguation phase, Melo, et al cites Amitay et al (2004) where they indicate that there are "two possible sources of ambiguity, namely geo/geo cases (for example London in England vs. London in Ontario) and geo/non-geo cases (e.g. the term London in Jack London, which should be considered as part of a name for a person)" [6]. According to Melo, et al, "In order to disambiguate place names, the surrounding words are analyzed" [6]. Using the surrounding words, there are phases associated with the disambiguation. First, if the place name has a surrounding word, such as a country or state, "a high confidence level is assigned to that place name" [6]. If there is not a match, then it is assigned a lower confidence and defaults to the location with the highest population. Finally, once the disambiguation step is complete, the algorithm finds the geographic region of the document. Using this system, the above algorithm "predicted the country that was the focus of each of these articles with an accuracy of 92% and an accuracy of 38% was measured for the task of correctly predicting the city that was the focus of each of these articles" [6]. While 92% is an excellent accuracy for determining the appropriate country, 38% accuracy for cities leaves much room for improvement. Melo et al next survey the "probabilistic language modeling approaches" that are in use as part of information retrieval algorithms. According to Melo, et al "By having language models for different geospatial regions, documents can be geocoded by choosing the model (i.e. the geospatial region) that best explains its textual contents" [6]. When discussing the best metrics in measuring the accuracy of the geocoders surveyed, Melo et al states, "It is particularly interesting to consider the median distance in the context of evaluating document geocoding methods, given that the median is relatively robust to outliers, and given that distances are easy to measure and probably easier to interpret than metrics of classification accuracy" [6]. While this is true of datasets with known locations -- for instance, modern web sites, newspapers, etc -- this is difficult for places where the exact location is unknown or impossible to measure (for instance, "Heaven" or "Atlantis"). Therefore, methods that rely on the spatial proximity of other locations may not be the best way to disambiguate ancient or historical texts.

Melo indicates that the field is ripe for further study and analysis. For instance, Melo et al states, "Despite the interesting results that have been reported in the articles surveyed here, there are also many possibilities for future improvements, as well as for the application of these ideas within subsequent tasks related to geospatial text mining and retrieval...Advances in document geocoding can be used to improve the resolution of individual place references or even to aid in the more general problems of named entity disambiguation and text semantification" [6]. Melo et al even demonstrate that multilingual geocoders are ripe for study when they state that, "...thus transferring knowledge from existing English corpora we can perhaps significantly improve the results of other languages" [6].

Continuing with the number of geographic references in the text, Zaila et al in *Geographic Information Extraction, Disambiguation and Ranking Techniques* state that “The toponym disambiguation algorithm is based on the assumption that in a document, the geographic location with more neighbors also present in the text has the higher probability to be the one referenced” [7]. Along these lines, Zaila et al describe four approaches to disambiguation, “knowledge-based, map-based, data-driven methods, or a combination of the three” [7]. According to Zaila et al, knowledge-based and data-driven approaches use “toponym-related information” discussed previously such as “population, administrative hierarchies or physical areas” [7]. Map-based approaches are what we see when using geometry to assign a probability, “often in the form of Euclidean distance” [7]. However, Zaila discusses the probabilistic approach mentioned above and “found that this new approach (theory of probability) was superior to most other approaches merely based on geographical scopes” [7]. This statement further indicates that a probabilistic approach would serve ancient and historical texts well in that it does not rely solely on a known location and proximity to each other, but rather on a probability that the locations are related. Furthermore, this research proposal will use a probabilistic approach to match/disambiguate places and persons.

To further stress the point on not relying solely on geographic proximity, Spitz et al give a great modern day example. Spitz et al state, “In other cases, however, spatial proximity is misleading. Consider, for example, the Canary Islands, which are located on the African continental plate in the Atlantic Ocean. Based on proximity, they are related to the countries of Morocco and Mauritania, to which they are closest. A more thorough investigation reveals, however, that they are an autonomous community of Spain and thus belong to the European

Union politically and culturally. Depending on the intended application, such a cultural or political relation can be preferable" [8]. Therefore, the current proposal will use tensors to track relations regardless of spatial location. For most ancient and historical texts, cultural and political connections are extremely important. Spitz et al use co-occurrence networks for their geocoding model. The current proposal will do the same, but instead of using graph theory, this proposal will expand the use of co-occurrence with tensors. Spitz et al states, "Disambiguation as a task stands to profit immensely from the availability of a network that represents the context of entities, as the context alone is often sufficient to disambiguate a term" [8]. They go on to state that the first step, which is mentioned in most articles, is to tag the place name, or toponym, "a task for which any named entity recognizer or gazetteer can be applied" [8]. According to Spitz et al, the tagging will result in three outcomes, "(i) no match is found, (ii) exactly one match is found, or (iii) more than one match is found (ambiguous mention)" [8]. It is the third option that our current proposal will use a tensor based solution to solve.

One group of researchers even go so far to to view the disambiguation technique as an enhancement to the identification step. According to Habib, et al, "A general principle in our work is our conviction that toponyms extraction and disambiguation are highly dependent" and that in previous research they studied "the potential of using the result of disambiguation to improve extraction" [9]. The authors term this "the reinforcement effect" [9]. Their approach is a cyclical method whereby they run the algorithm to find candidates, disambiguate those results, then run the algorithm again. They repeat this process until there are no more candidates with ambiguous results. According to the authors, their "... [Reinforcement Effect] approach managed to improve precision through iteration of refinement" [9]. In a another paper by Habib, et al, the authors go on to state that they use a Hidden Markov Model to extract the toponyms and then run the toponym candidates through a Support Vector Machine to categorize the toponym candidates as “false positive and true positive toponyms” [10]. In yet another article by the same authors, they discuss using three different approaches: 1) a Naive Bayes approach which “is a probabilistic approach to text classification” [11]. 2) a “Popularity Approach” where “... the more of those [toponyms] appear in a certain country, the more probable it is that the document belongs to that country” [11]. And 3) a “Clustering Approach” which states that “...toponyms appearing in the same document are likely to refer to locations close to each other distance-wise” [11]. This research may prove a promising method to consider in this proposal, especially during the “Identifying Potential Candidates” phase, the first phase in our proposed method.

Moncola, et al demonstrate that there are four types of ambiguity in a text: 1) referent ambiguity, which “the same name used for several places;” 2) reference ambiguity which, “the same place can have several names;” 3) referent class ambiguity, which “the place name can be used in a non-geographical context, like organizations of names of people;” and 4) structural ambiguity, which is “when a structure of the words constituting the place name in the text are ambiguous (e.g is the word *Lake* part of the toponym *Lake Grallaleu* or not?)” [12]. A method of interest to the current proposal is how the authors deal with referent ambiguity. The authors use clustering algorithms to determine the relationships between the candidates. Specifically, the candidates use the DBSCAN (Density-Based Spatial Clustering of Algorithms with Noise), developed by Dr. Xiaowei Xu, et al: the advisor for the current research [13]. DBSCAN allows for the authors to form neighborhoods of candidates and allows for them to account for outliers,as well [12]. This method may prove a useful supplement to tensors in the current research proposal.

Furthermore, Wolf, et al in *Characterization of Toponym Usages in Texts* not only lay out types of disambiguation, but various types of toponyms one should consider [14]. They discuss the following linguistic devices: A) Usage Kinds, and; B) Stylistic Devices. The authors state that one usage kind is “Reference” [14]. An example of a reference is when one uses prepositions of postpositions with a toponym. Another usage kind is “Attribute” [14]. Attributes are when toponyms are used to describe other nouns located in the sentence such as an adjective, possessive, etc. For example, the authors use a possessive example of “Iceland’s POLICE denies its responsibility” [14]. Here, Iceland is indeed a toponym, but it is being used adjectivally to describe which police department. The third and final usage kind is called “Standard” [14]. This corresponds to the general usage of a toponym to describe a location and are typically used with adjectives. The authors go on to list two categories -- Figures and Tropes -- and ten sub categories for “Stylistic Devices” [14]. While we will not go into detail in this proposal, the following Stylistic Devices are mentioned and worth considering in our algorithm: Figures include Alliteration, Geminatio, Anaphora, Epiphora, Anadiplosis, and Comparison. Tropes include Synecdoche, Antonomasia, Metonymy, Metaphor, and Allegory [14]. Each of these categories and subcategories may prove useful in our research.

Another avenue that may prove promising to our research is the previously mentioned idea of co-occurrence. Zhong, et al in their article *Analysis of co-occurrence toponyms in web pages based on complex networks* proposes “... a novel complex network model for co-occurrence toponyms…” [15]. The authors demonstrate that co-occurrence can be a viable method for identifying candidates or what many call extraction. For instance, the authors go on to state that it is not just the co-occurrence of toponyms in one text that can determine relationships between toponyms, but "...documents can link to each other through shared toponyms" [15]. The authors use these two co-occurrence methods to create direct links (toponyms within the same text) and indirect links (toponyms found in different texts) between vertices. According to the authors, "...by studying the transition effect between co-occurring toponyms, we iteratively calculate the importance value of a vertex based on link analysis, then identify the significant toponyms with remarkable features or navigation functions in huge network resources, which can be applied to geographical information retrieval" [15].

Named Entity Resolution and Disambiguation

Of course, co-occurrence is not just used for geocoders, but for all forms of entity resolution. [16]. Eckhardt, et al discusses candidate matching and disambiguation in *Entity Linking Based on the Co-occurrence Graph and Entity Probability* [16]. As the title depicts, the authors use co-occurrence and graph theory combined with probability theory to match and disambiguate entities in text. However, the authors are not just searching for toponyms, but any entity found in the text . This is very important for this research, for this article does not discount persons or other nouns in the disambiguation process. However, it does differ in that it does not "geocode" the results, but uses toponyms to determine which entities are connected. Also, this article demonstrates the use of graph theory and probability to determine the connections where our research proposes to use tensors in order to determine a probability between candidates [16].

When researching the current state of the field of Named Entity Recognition (NER) and Named Entity Linking/Disambiguation (NEL/D), there does not exist a lot of data on domain specific Named Entity Recognition/Disambiguation tools and analysis. The research is heavily focused on using what is known as “big data” resources to analyze and tag. For instance, most research involves large social media platforms such as Twitter or news organizations for the raw text and Wikipedia and linked open data (RDF specifically) for the knowledge base. Using graph theory or clustering, they score the candidates for matching. While there is overlap in this research and the current research, this is a bit like comparing apples and oranges. For one, this research and tool is designed for large corporas of texts -- hundreds and possibly thousands of texts dedicated to a specific resource domain, for instance, late antiquity -- while their research is dedicated to finding contemporary entities in “big data” datasets. These large knowledge bases do not contain the granularity of entity data necessary for domains such as studying early Christian writings in late antiquity. To run our datasets through these tools and compare scores would be to do a disservice to their research since they would end up with numerous false negatives -- their knowledge base just would not have our entities in them -- and false positives since they may have an entity with the same name, but not even be from the same place or even time period.

However, there is merit in looking at the processes these research groups use in Named Entity Recognition and Disambiguation as well as looking at their benchmark scores. On the one hand, some salient points on their process can influence and assist in the current process of disambiguation. On the other hand, comparing the scores will also demonstrate how a domain specific Named Entity Recognition/Disambiguation tool will greatly assist the end user who may deal in a restricted domain in size, space, and time.

The current state of research relies highly on graph theory for Named Entity Recognition and Disambiguation. For instance, Wiener, et al developed a tool that reads news articles each day and relies "on a World Knowledge Graph derived from data sources such as Wikipedia to represent readers' a priori domains knowledge, and a Document Graph to represent the article at hand…" [17]. The authors state that all possible entities, as well as "inferred article categories" containing different categories or topics for the document, and term frequencies are added. They then develop a subgraph containing entity nodes, category nodes, stop words, etc. From there, they also create edges between each of the nodes based on their connections and contain weights derived from the original graph, in this case Wikipedia. In the first step of the process "graph nodes are weighted according to the sum of their edges weights" [17]. From there, they have two groups, "solved" and "ambiguous" or what they call "Strongly Connected Components [(SCC)]" [19-13c]. Here, "[entity] scores are based on their edges to each of the solved group SCCs and their weights…" [17]. This is very similar to the current project in that both projects use "solved" or "certain" entities to calculate the scores for the ambiguous entities. However, their calculation uses weighted edges between nodes where we use a Bayesian scoring system by counting connections in the tensor for every text. One item in this research to note for future research is that the authors' goal of this article is to answer how to handle "empty entries" or what the current paper calls "unknown entities". Unfortunately, the authors do not give any measurements such as Precision, Recall, or F-score for comparison.

Another set of research is presented by Zhang et al called “AMiner” which “is a free online academic search and mining system…” [18]. The goal of their research is to “present the implementation and deployment of name disambiguation, a core component of AMiner” [18]. According to the author’s abstract, “... [name disambiguation] has been studied for decades but remains largely unsolved” [18]. There is one notable difference between the authors’ research and the current research: Zhang et al add a “human annotators” into the disambiguation process. According to the authors, “Name disambiguation is usually viewed as a clustering problem… The state-of-the-art solutions for name disambiguation problem can be divided into two categories: feature-based and linkage-based” [18].

Feature-based Methods group the candidates then use Support Vector Machines (SVM) and “finally employs DBSCAN to cluster documents” [18]. Using these algorithms, one can see which cluster the candidate falls into and can make a determination. The second method, Linkage-based, are “methods capable of utilizing graph topology and aggregate information from neighbors” [18]. Many in this method use Hidden Markov Random Fields [19] to create their network and determine probability. While the current research would fall into the “linkage-based” methods, the current research uses a tensor based approach and Bayesian probability to determine the candidate match. Zhang, et al complete their process using clustering to determine the match. Along with Hidden Markov approach, others are those using Latent Dirichlet Allocation (LDA) [20], which is in fact a Bayesian probabilistic model using prior knowledge, similar to the current research. However, the current research is not looking at the full-text for topics in documents to determine the probability, but the proximity of words co-occurring with each other. For instance, one group of researchers use Mallet topic modeling tool to create topics in use for disambiguation [21, 22]. This is using a Bayes approach, but in a quite different way from the current method.

One of the biggest differences here lies in the fact that they are using documents to determine disambiguation in a global embedding functions, which are based on supervised connections. However, what about if two of the same names occur in the same document? Clustering is difficult in this situation. An example case one will see from the current research in a subsequent section: how would you handle three different people named “Mary” if they are mentioned in the same text? Or same paragraph? Or even the same sentence? The authors’ then supplement the global embedding functions with “local linkage learning” which uses an unsupervised graph for each candidate.

The authors’ tested a sample of 100 names and benchmarked it against four other methods. The results of the authors’ experiment yielded the following averages for the 100 sample names:

Precision	**Recall**	**F1**
77.96	63.03	67.79

Table 1: Results of Author Name Disambiguation from Zhang, et al

As one can see, this is a sampling of 100 names out of a possible 12,798 authors. This sample represents around 0.7% of the total number of authors. The results demonstrate an F-Score of 67.79, which was a considerable improvement over the other four benchmarks presented by the authors: 62.81, 50.23, 63.10, and 53.42 [18].

Another set of researchers, Duan, et al, uses co-occurrence for disambiguation. In their research, they break the field down into two categories: "1) One is knowledgebase-aware to solve name (also called a mention) ambiguity problem… also called entity linking; 2) The other focus on the features of context in respective enclosing document of ambiguous names, also called entity clustering" [23]. The authors also note that machine learning is making its way into the field of disambiguation. However, the authors' conclusion on using machine learning is "(t)he main drawback of the machine learning method is that the performance of the algorithm is limited by the quantity and range of the training corpus" [23].

In the authors' research, they decide to forgo the need of a knowledge base and use a clustering technique. Not only do they use a clustering technique, but they also use a "manually annotated dataset" instead of a public dataset. This is the same approach as the current research in this document. For English, they use 471 documents containing 16 names and 97 entities. Their results yield the following:

Precision	**Recall**	**F1**
0.833	0.947	0.886

Table 2: Results of Duan, et al research

The authors also state that, "(o)ur approach can work under a low system resource requirement and achieve a good performance" [23]. It is uncertain how big a "document" is, but according to their chart they can process a million documents with under 2.5 GB of memory.

However, the information from their article indicate that these are very small documents. For instance, 471 documents in English yield only 16 names and 97 entities [23].

An instance of a multilingual approach is found in research by Moussalem, et al in *MAG: Multilingual, Knowledge-base Agnostic and Deterministic Entity Linking Approach*. One of the most salient notes in the article is, “However, our experiments … show that the underlying models being trained on English corpora make them prone to failure when migrated to a different language” [24]. Another feature that can be useful in future implementations of the current research is the fact that they use n-grams (specifically trigram) and stemming -- although the current research does use stemming/lemmatization when tagging Greek -- when analyzing text. The authors’ provide an example, “...MAG stems ‘Northern India’ to ‘North India’ to account for linguistic variability [24]. They also use TF-IDF in their algorithm. This method was considered in the current research. However, there are many instances of ambiguous names occurring in the same text, paragraph, and sentence. Therefore, TF-IDF did not appear as useful in the domain in question. MAG then uses a directed graph and a Breadth-First Search technique to find hidden paths. At this point, they use what is known as HITS and Page Rank. “HITS uses hub and authority scores to define a recursive relationship between nodes” [24]. Also, like other researchers mentioned above, if they cannot determine any candidate, then the entity is called an “Emergent Entity” [24].

The authors used various datasets ranging from news, tweets, etc. The following are Micro F-measures for their algorithm using either HITS or Page Rank, plus twelve other benchmark algorithms:

AGDISTIS	AIDA	Babelfly	DBpedia	DoSer	entity classifier.eu	FRED	Kea	NERD-ML	PBOH	WAT	xLisa	MAG+HITS	MAG PR
0.45	0.48	0.50	0.36	0.55	0.24	0.11	0.53	0.31	0.59	0.54	0.45	0.63	0.59

Table 3: Results of Moussalem, et al research

One of the tools mentioned in the preceding research, as seen in Table 3, is FRED. FRED, designed by Gangemi, et al and described in *Semantic Web Machine Reading with FRED*, is "a machine reader for the semantic web: its output is a RDF/OWL graph…" [25]. FRED uses Entity Linking and the output of RDF/OWL so that machines can read and interpret the data. This is similar to the current research in that the process will output a TEI/XML file for use by machines. However, the TEI/XML file will not contain the linking information; this is stored in the tensor.

There are also tools, such as GERBIL, that aim to combine multiple knowledge bases and annotators into one product. In *GERBIL- Benchmarking Named Entity Recognition and Linking Consistently* by Roder, et al, the authors aim to create a system that will allow different researchers the ability to test using the same datasets, etc [26, 27]. The authors state the exact scenario described at the beginning of this section when they state, "(t)he issue of comparability of results is not to be regarded as intrinsic to the annotation task. Indeed, it is now well established that scientists spend between 60% to 80% of their time preparing data for experiments" [26]. The authors state that this leads to the following: "(t)hese restrictions have led authors evaluating their approaches on datasets (1) that are available to them and (2) for which writing a parser and an evaluation tool can be carried out with reasonable effort" [26]. The authors also state that there is not a unified approach to measurement, either. For instance:

> ...while some authors publish macro-F-measures and simply call them F-measures, others publish micro-F-measures for the same purpose, leading to significant discrepancies across the scores… This heterogeneous landscape of tools, datasets, and measures leads to poor repeatability of experiments, which makes the evaluation of the real performance of novel approaches against the state-of-the-art difficult [26].

While GERBIL is a noble effort in solving this issue, it still relies on very specific formats and datasets that are irrelevant to many domain specific research interests, such as early Christian writings in late-antiquity. This finally leads us to two different research projects that deal with specific corpora of texts instead of "big data" scale projects using DBpedia, etc.

Using GERBIL, Munnelly et al test the tools combined in GERBIL to see how well they perform "... the task of disambiguating entities in the 17th century depositions obtained during the 1641 Irish Rebellion" [28]. Written in June 2018, the authors give a blunt conclusion right in their abstract:

> Based on this corpus and the results obtained from the General Entity Annotator Benchmarking Framework [GERBIL] we observe that the accuracy of existing Entity Linking systems is limited when applied to content like these depositions. This is due to a number of issues ranging from problems with existing state-of-the-art systems to poor representation of historic entities in modern knowledge bases [28].

The authors did not create their own knowledge base for this research. In fact, they are upfront in stating that, "(w)e will also continue to seek answers to the problem of dealing with poor representations of entities in knowledge bases. This is undeniably our greatest challenge" [28]. The current research will greatly assist them in this task since the goal is to provide a clean and simple way for one to create a domain specific knowledge base. Another salient note about GERBIL mentioned in the article is that "...some of the [entity linking] systems provided by GERBIL cannot perform NER…" which means that they cannot perform analysis on raw text; they can only work if a human annotates it before running the process. This is a considerable drawback for the process [28].

The results of Munnelly, et al for their dataset using GERBIL are as follows:

Annotator	**Macro F1**	**Macro Precision**	**Macro Recall**	**Micro F1**	**Micro Precision**	**Micro Recall**
AGDISTIS	0.5979	0.5979	0.5979	0.6052	0.6052	0.6052
Babelfly	0.1299	0.2941	0.0833	0.1130	0.3348	0.0743
DBpedia Spotlight	0.1449	0.4767	0.0854	0.1281	0.4970	0.0774
Dexter	0.1082	0.3333	0.0646	0.0933	0.3536	0.0580
FOX	0.4051	0.6327	0.2979	0.4054	0.6791	0.2999
FREME NER	0.1012	0.3118	0.0604	0.1045	0.3076	0.0694
KEA	0.1466	0.3358	0.0938	0.1363	0.3384	0.0923
PBOH	0.4250	0.4250	0.4250	0.4266	0.4266	0.4266

Table 4: Results of Munnelly, et al analysis

As mentioned previously, the authors provide a blunt conclusion that the state-of-the-art methods are still not good enough to provide the accuracy needed by domain specific projects. This project is very similar to the current research in that the time period and the entities are missing in the large open knowledge bases [28].

In NEREA: Named Entity Recognition and Disambiguation Exploiting Local Document Repositories, authors Garrido, et al demonstrate a process for Named Entity Recognition and Disambiguation for local repositories, which is much closer to the current research, but still quite different. Here, the authors use TF-IDF and TF-WP for each word then builds a vector based on bag-of-words. To determine the candidate, the process uses cosine similarity of the vectors to determine the correct word. This process further differs from the current research in that it uses both a global knowledge base (DBpedia) and a local knowledge base. However, the authors however did not use a manual dataset, but a publicly available dataset that is already annotated. This allows them to compare their experiment with other processes, as well as use the global knowledgebase. The following are the comparison results:

Tool	**Precision**	**Recall**	**F1%**
Adel-w/o pruning	49.4	46.6	48
Adel-w/ pruning	57.9	6.2	11.1
AIDA	51.6	43.9	47.4
TagMe	28.5	54.9	37.5
DBpedia	28.3	45.7	34.9
Another research group	73.1	46.1	56.5
NEREA	79.4	59.5	68

Table 5: Results of Garrido, et al research using various tools

After completing the experiment on publicly available data, the authors then run the process on a local dataset using a custom created knowledgebase. To create the knowledge base, the authors manually harvested all entities in the news published in 2014 and 2015. They then randomly selected 5,000 ambiguous references and human verified the results. They first ran the experiment using a global knowledgebase, DBpedia. Following that, the authors then ran the datasets using both global and local knowledge base. The results are as follows [29]:

Knowledgebase	**Precision**	**Recall**	**F1%**
Global KB Only	55.2	45.3	49.7
NEREA	83.5	62.4	71.4

Table 6: Results of Garrido, et al research using custom tool

The authors demonstrate the concept discussed earlier in the current research: a locally created knowledgebase is superior for domain specific projects than a global knowledge base such as DBpedia, etc.

Finally, Lasek, et al in their journal article *Various approaches to text representation for named entity disambiguation* make the argument that structured co-occurrence models provide the best results in not only English, but other languages. In their article, the authors use a co0occurence within the same paragraph to make connections. Their dataset used in the experiment was rather small: only "..nine news articles with 106 entities…" [30]. According to the authors, the co-occurrence method provided a 91.9% Precision and above 70% Recall. While the authors' algorithm differs in many ways, this demonstrates the effectiveness of a co-occurrence model [30].

Along with this co-occurrence approach, the current research strives to use a domain-specific knowledge base. This approach is also encouraged by Zhang et al in their article *A Two-Stage Joint Model for Domain-Specific Entity Detection and Linking Leveraging an Unlabeled Corpus*. In the article, the authors demonstrate the use of n-grams and a combination of global and local knowledge bases. However, when using co-occurrence, the authors assume that each document is a topic and therefore an entity mentioned in the document belongs to said topic. As mentioned previously, this works well for most scenarios, but it does not work well when multiple people with the same name are mentioned in the same document, as is the case in our corpus. Nevertheless, the authors indicate that they are able to achieve the following overall scores [31]:

Precision	**Recall**	**F1**
81.18	68.43	74.26

Table 7: Results of Zhang, et al research

The authors have demonstrated that a local knowledge base works well given domain specific requirements over just using a global knowledgebase.

Research Approach

Novelty

This project takes the novel approach of using a rank-3 tensor to disambiguate entities. Most other algorithms in Named Entity Resolution/Disambiguation (NER/NED) use a static knowledge base to match candidates and disambiguate. (see State of the Topic section above). Also, many of the current state-of-the-art disambiguation tools require annotated text; they cannot even perform named entity recognition on raw text. However, this approach tries to approximate how a human would read a source, thereby adding knowledge to the knowledge base -- or learning -- as it reads and sometimes reading multiple times to supplement earlier entities that were still too ambiguous. Each text is added to the tensor where it is available for future processing of texts in the corpora.

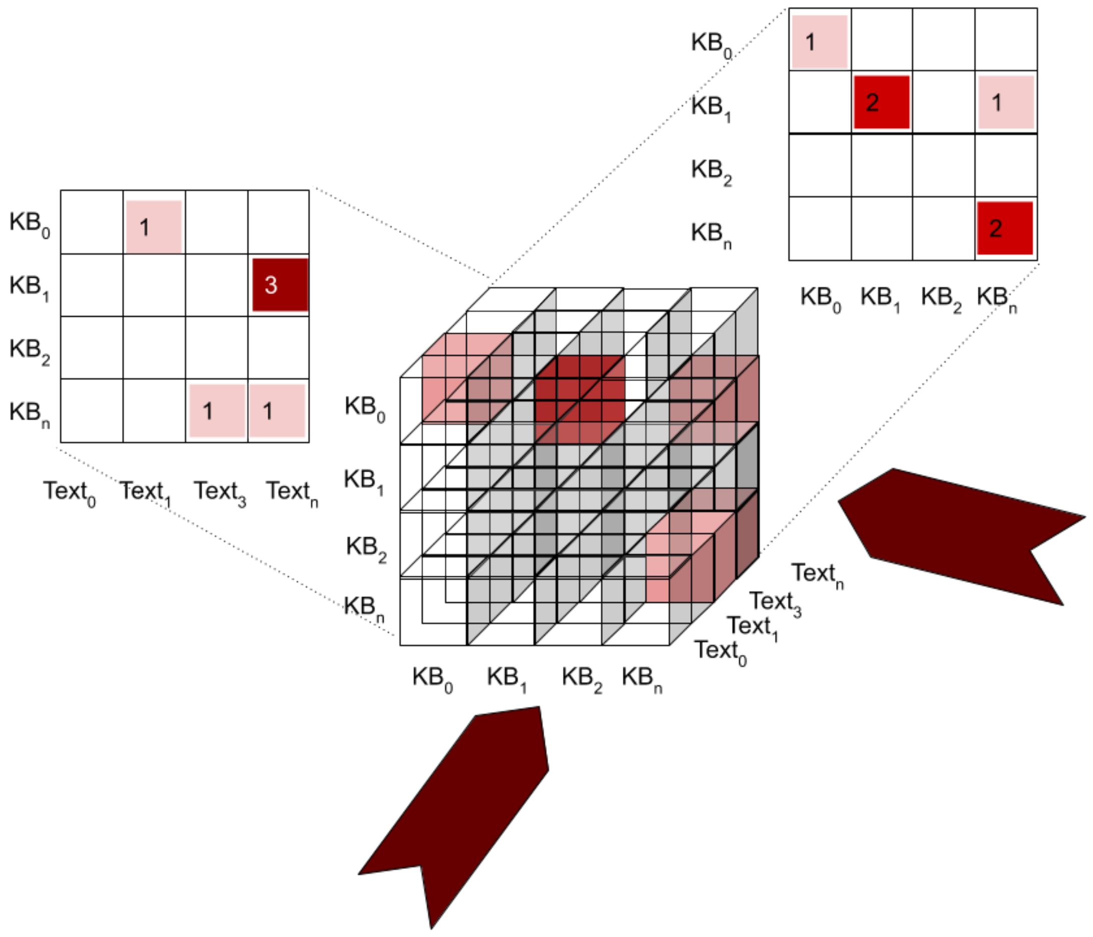


Figure 1: A diagram depicting the rank-3 tensor of Knowledge Base Entities (KB) and Texts 0-n

Furthermore, this approach uses a very simple knowledge base. A scholar need only input a name, surname (if available), some aliases (if available), and a few connections for this program to tag. Most current scholarship uses linked data -- typically RDF files from Wikipedia, DBpedia, etc -- as its knowledge base. Using RDF with the SPARQL query language on the "semantic web", one has the ability to analyze and connect this linked open data [32]. This is similar to how the current research uses TEI and xQuery for the same analytical capabilities. According to recent articles, using RDF and SPARQL works well for representing tweets, newspaper articles, etc. However, as one sees in "Current State of Research" section, these current knowledge base sources are inadequate for the domain of Christian sources of late antiquity. Using this simple knowledge base solution combined with the tensor approach results in a more accurate tagging of these sources. That being said, a future component of research includes using an RDF standard as a possible choice for knowledge base.

Once the knowledge base is complete, the program turns each entity into an object and stores it for processing and comparison. Once possible candidates are matched in the entity matching process, the objects rely totally on the numeric values assigned. Using these numeric values, a Bayesian probability approach is used to determine which candidate is the actual match given not only the text, but the entity's location in the text.

This approach also produces an output that is beneficial to scholars in the digital humanities field. Using an encoding standard known as Text Encoding Initiative, or TEI, the code produces Extensible Markup Language, or XML, files that are uploaded to no-sql databases such as eXist-DB for analysis. Using this standard allows for anyone familiar with TEI to quickly perform their own analysis using xQuery, translate the data into another format using xpath, or even incorporate the data into their own project. [33, 34, 35].

Benefit to Industry/Field

While one of the goals of this research is to develop a digital humanities machine reading tool that will tag persons and places in early Christian texts, there are implications for any NER/NED algorithm. Many organizations or researchers -- such as an archive or library -- may have a specific domain where they need a customized knowledge base to search, match, and tag entities. On top of that, they may not want to take the time to create RDF files or complicated relational databases for their project. Given this tool, one needs only to create a comma delimited file knowledge base with some connections and they can achieve a high level of confidence in their automated tagging.

Existing tools may look at the concepts of adding knowledge from previous texts and as the algorithm processes the text in order to increase their matched candidates. Assuming that the knowledge base contains all of the connections needed misses out on the benefits that comes when a machine can "learn" with each match. Also, others may look at the various aspects such as co-occurrence distance calculated based on average sentence length, growth factor in co-occurrence based on a percentage of the text size, etc.

Named Entity Recognition Process

Domain Specific Knowledge Base/Authority File

The first step is to incorporate a knowledge base for processing the text to find entities. One may think of the knowledge base as the “brain” of the computer program. The knowledge base contains the names of entities -- in our case people and places -- and connections among the various entities. The goal of this research is to demonstrate how a simple domain specific knowledge base, when combined with a tensor, can tag entities in raw text files with a high level of accuracy.

In order to keep the process as simple and quick as possible for any scholar, a simple comma delimited file is used. The program requires the following fields at a minimum: unique ID, Name, Description, Surname, Alias$_1$...Alias$_n$, Connection$_1$...Connection . When using the Greek functionality, one must also add the following: GreekName, GreekSurname, GreekAlias$_1$… GreekAlias$_n$. Please note that the connections do not change when using English and Greek. As one will see in the “Entity Match” section, once the values are in the system, the tokenized words and the knowledge base entities are converted to numeric values. Therefore, the connection in English between Entity A and Entity B is the same in both languages.

Entity Name

When updating the name in the knowledge base, one may use either the first name of the individual or the surname of the individual. The algorithm is designed so that both are interchangeable. The reason for this is found in the intricacies of name use in late antiquity. For instance, Mary Magdalene can easily be a name/surname combination. However, Joseph of

Arimathaea is not always found in the Name: Joseph, Surname: Arimathaea combination that one sees in Mark 15:43. We see references to him as, "... a rich man from Arimathaea, named Joseph" [Matthew 27:57] or the even more difficult "a man named Joseph, a counsellor; and he was a good man and a just (the same had not consented to the counsel and deed of them;) he was of Arimathaea" [Luke 23:50-51]. Furthermore, take the disciple most modern people would easily recognize by the name Peter. In the gospel narrative, Peter is actually Simon son of Jona, but is frequently called Simon Peter. Therefore, one may use Name: Peter, Surname: Simon or Name: Simon, Surname: Peter, with numerous aliases and a connection to Jona.

The above are just a few examples of how one may setup the knowledge base. Notwithstanding these intricacies, the algorithm is designed to handle a variety of name issues in order to match candidates then disambiguate them regardless of the choices made in loading data into the knowledge base.

Supervised Portion: Connections and Seeding

While this research uses a simple knowledge base structure, like other algorithms in the field, it uses supervised connections to aid in disambiguation. Other algorithms appear to use a format known as RDF

```
<rdf:RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#" ...
<skos:Concept rdf:about="http://syriaca.org/person/8">
      <rdf:type rdf:resource="http://lawd.info/ontology/Person"/>
      <rdfs:label xml:lang="en">Miles , bishop of Sus</rdfs:label>
      <rdfs:label xml:lang="syr">ܡܝܠܣ</rdfs:label>
      <lawd:hasName>
         <rdf:Description>
            <lawd:primaryForm xml:lang="en">Miles , bishop of Sus</lawd:primaryForm>
         </rdf:Description>
      </lawd:hasName>
      <lawd:hasName>
         <rdf:Description>
```

```
        <lawd:variantForm xml:lang="en">Miles , bishop of Sus</lawd:variantForm>
        <lawd:hasAttestation rdf:resource="http://syriaca.org/bibl/4"/>
      </rdf:Description>
    </lawd:hasName>
...
```
Figure 2: An example of a portion of an RDF file for person 8 [36].

As mentioned earlier, the use of linked data and RDF is a future avenue of research to add to the current algorithm described in this research. However, the current goal is to: 1) allow for a scholar to create a simple knowledge base without the need to learn or design an RDF encoding standard for their corpora while 2) demonstrating the accuracy of the tensor method.

The term "supervised" in machine learning refers to the component pieces where a human is involved in the process. In this research, the only supervision is in the seeding of our knowledge base. Essentially, a scholar must input that Entity 123 is connected to Entity 678. These simple connections in the comma delimited knowledge base is all that is needed for the algorithm to analyze the texts.

Training Data

For the purposes of our research, the gospels of Matthew, Mark, Luke, and John found in the King James Version Bible hosted in the Gutenberg project were used to test and design the algorithm [37, 38, 39, 40]. For the Greek training and testing, a Greek version of the gospel of Matthew is used [41]. One important note regarding the Greek text: there was some pre-processing steps taken to clean the data before running the algorithm.

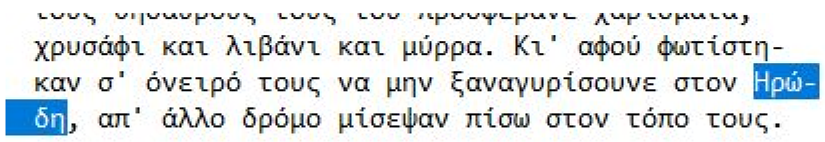

τους θησαυρούς τους του προσφέρανε χαρίσματα,
χρυσάφι και λιβάνι και μύρρα. Κι' αφού φωτίστη-
καν σ' όνειρό τους να μην ξαναγυρίσουνε στον Ηρώ-
δη, απ' άλλο δρόμο μίσεψαν πίσω στον τόπο τους.

Figure 3: Example of preprocessing required on Greek text

These documents are the perfect sources for this task given that: 1) the sources are primary sources from the time period and genre; 2) each gospel gives a narrative of similar events, people, and places; 3) each narrative, while similar, uses different names, different structure, different stories, etc, and; 4) provide a way to train on three texts and test the fourth. Therefore, this research used the algorithm on multiple runs through these gospels to tweak the algorithm for the best results and to handle the intricacies mentioned above. Through this process, even a human accustomed to reading these sources had to consult documents to see which entity is the actual entity mentioned in the text, a book from 1893 titled *Illustrated Bible Dictionary and Treasury of Biblical History, Biography, Geography, Doctrine, and Literature* by M.G. Easton was used for supervised human disambiguation only [42].

The use of this source brings up the question, "why the use of a text from 1893 instead of a current text?" The answer lies in modern copyright law. Since the author did not intend to just quote a couple lines from a source, but to use it extensively throughout the knowledge base. While there are many good modern books such as the *The Oxford Guide to People & Places of the Bible* by Metzger and Packer [43], the author made the determination to stick with sources that were out of copyright and could be used freely so that the code, knowledge base, etc for this research can be made available to the public free of charge. However, please note that some resources such as the previous book were used as a resource only when clarifying ambiguity of some entities. The only data or texts used in this research -- other than researching other algorithms in the field noted in the "Current State of Research" and sourced throughout -- is known as the Apophthegmata Patrum [44]. This translation is still under copyright. However, permission was obtained from the publisher Cistercian Publishing to use the raw text for analysis and to publish the results. However, permission was neither requested, nor granted for publishing the complete tagged text.

Testing Data

After seeding the knowledge base with entities found in the gospels, an index from the Ante-Nicene Fathers housed digitally at Calvin College. As mentioned in Calvin College's summary of the text, "Originally printed in 1885, the ten-volume set, Ante-Nicene Fathers, brings together the work of early Christian thinkers. In particular, it brings together the writings of the early Church fathers prior to the fourth century Nicene Creed" [45]. Again, careful consideration was made to use only texts and data that were out of copyright. One may follow up the earlier question with, "would not the use of modern scholarship not aid in the results of this research?" While modern scholarship may elucidate some minor aspects of these texts, one must remember that the scholarship surrounding these Christian texts from late antiquity (approximately 1AD-400AD) have been studied by scholars for almost 2 millennia. The vast majority of people/places mentioned in these texts have been agreed upon for most of that time period. While authorship, provenance, date, etc are open to debate, the specific entities are pretty static over time.

Step One A: Creating the Tensor on First Run

On the first run, one must load the above knowledge base into the rank-3 tensor using the loadBulk() function found in the TensorBulkLoad class. This research uses the Python numpy package to create the rank-three tensor a[x,y,z] [46, 47], . The variables x and y must be the size of the maximum ID value of the knowledge base. The variable z corresponds to the number of texts that the tensor will hold. Therefore, the tensor is defined as:

a[*KB maximum ID value*, *KB maximum ID value, Number of Texts]*

The process then reads in the csv knowledge base file and for each row adds all attributes from the knowledge base into an array called bulkArray[]. Once complete, the process uses the hickle library to dump/save the tensor onto the computer for persistent storage, but returns the bulkArray in memory for use in the current run [48].

On subsequent runs after the initial bulk creation of the tensor, the process simply reads the tensor from file with values from all previous text runs. Again, the hickle library is used to load the tensor called a[x,y,z]. This tensor "a" is used throughout the remainder of the program and is dumped/saved at completion and ready to use for any subsequent text runs.

Step One B: Determining Maximum ID in the Knowledge Base

From this point forward, the process is completely unsupervised, meaning there is no human interaction with the process. The first step is a very simple step, but a crucial component needed for future functions throughout the process: what is the maximum ID number found in the knowledge base? The function connection() connects to the knowledge base .csv file, loops through the knowledge base appending each ID into a list row-by-row. After the process loops through each row, the max() function is used to determine the maximum ID. The function then returns this number so that future functions have access to it. This number is crucial in many of the future processes as it is needed to loop through each entity in the knowledge base to see if it is a match. Once the connection() function completes, it will return the maximum ID value found in the knowledge base for future use in looping through the array of objects.

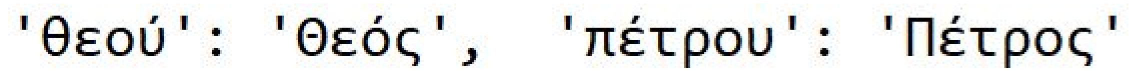
Step Two: Processing the Match and Loading Results into a TextToken Object

Read/Tokenize Raw Text

The second step of the program is to read the raw text files using a method called readText(). The program is designed to read texts stored in a variety of locations and formats. Option One: the process may read raw text provided as a URL to a website location, such as Gutenberg. Option Two: the process may read raw text provided as a text file stored locally on the network. Option Three: the process may read raw text input directly from the user. Furthermore, options one and two may be standard a standard text file, such as .txt, or it may also be a PDF file thanks to the incorporation of the PyPDF2 libraries [49].

Depending on if the text is English or Greek, the process will use one of two opensource libraries to tokenize the raw text. If the text is English, then the Python package Natural Language Toolkit (NLTK) is used. NLTK is a standard Python natural language processing (NLP) "suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries [55]. For the purposes of this research, only the "tokenize" function is used during the reading of the texts. This process creates a unique token for every word, symbol, punctuation mark, and even creates an object for "'s" when possessive nouns are used. The readText() function then places each token into a Python list returns the list called “TextList” to be used by future functions. Before doing so, the process makes sure to encode the text in standard UTF-8 format to make sure that there are no mismatches in the future processing due to format issues.

If the text is in Greek, the algorithm is essentially the same as above, but the process uses the Python Classical Language Toolkit (CLTK) for processing. CLTK performs many of the same functions as NLTK, but adds “natural language processing (NLP) support for the languages of Ancient, Classical, and Medieval Eurasia” [50]. The most complete languages available through CLTK are “Greek, Latin, Akkadian, and the Germanic languages…” [50]. For the purposes of this research of creating a program to analyze early Christian texts of late-antiquity, Greek and Latin are the justifications for using CLTK libraries in Python. For Greek, the process mirrors the English with one exception: lemmatization. The biggest difference between Greek -- and many other languages -- and English for the purposes of tokenization and matching is that Greek nouns change form based on different parts of speech. A greek word has a specific stem, but the ending takes on many different forms based on how the word is being used. There are two approaches to this: 1) one may stem the word and stem the knowledge base entity names and then match, or 2) one may use lemmatization and convert any variation of a noun to it’s nominative first person singular form. For the purposes of this research, the lemmatization method is used to convert words into the nominative singular lexical form.



```
'θεού': 'Θεός',  'πέτρου': 'Πέτρος'
```

Figure 4: An example of two words in the lemmata file going from Genitive to Nominative

CLTK comes with a lemmatization function known as “LemmaReplacer”. While the LemmaReplacer() function may work well for most Greek tasks, it did not work well out of the box for the early Christian corpora. Many of the proper nouns found in the gospels were not found in the lemma file. Therefore, many of the proper noun lexical forms found in the gospels had to be manually inserted into the CLTK lemma file along with the proper nominative singular lexical form found in the knowledge base. This provided the ability for the forms found in the raw text to be converted into the proper form for the future matching process.

Step Three: Entity Match

The third step, the entityMatch() function, is responsible for loading the knowledge base into an array of objects as well as creating an array of token objects for our tokens created in step two.

Creating AuthTokens for our Knowledge Base Entities

Using python, the program opens the comma delimited (.csv) file and appends each row of data into an object stored in an array (or a list in python terminology, but I will use the generic computer science term array and will not dive into the differences in this article). The object has the following elements stored in a class called “AuthTokens”:

AuthToken Class

ID: the unique identification number that the user assigned to the entity

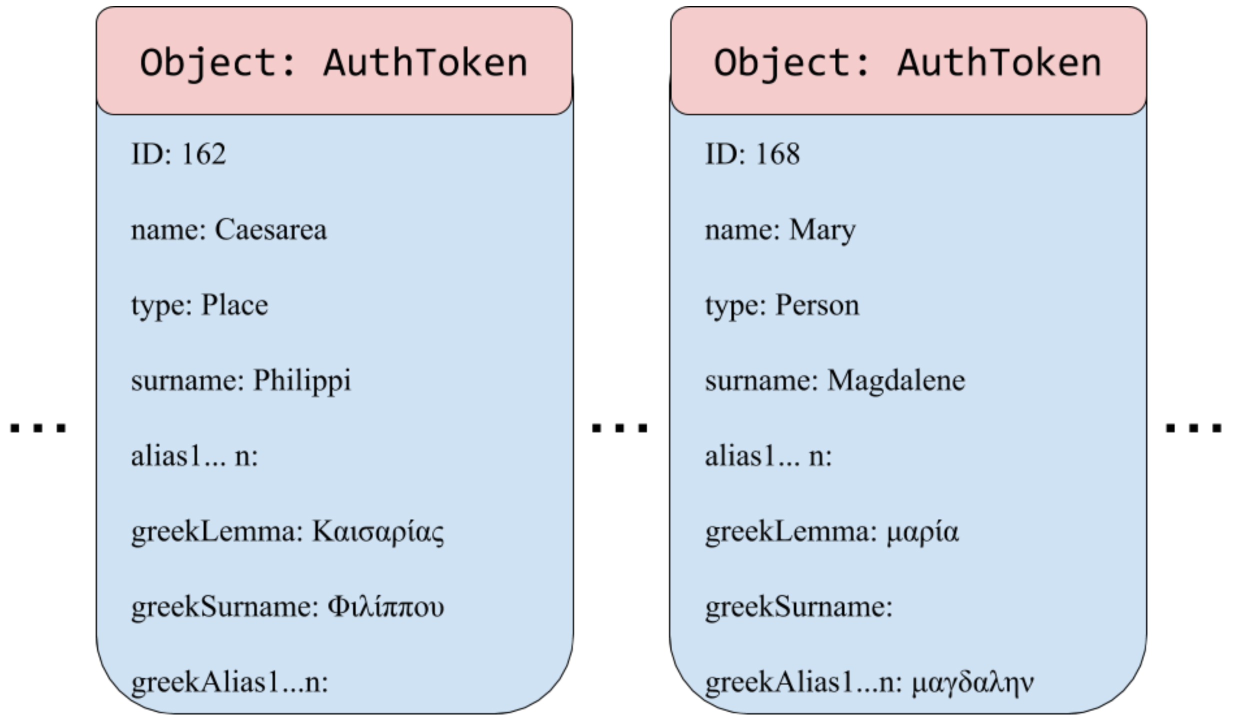
name: the name row of the knowledge base file (KB)

type: the type row of the knowledge base file (ie: Person, Place, Organization)

surname: the surname row of the knowledge base file

alias1… aliasn: the alias row(s) of the knowledge base file *greekLemma:* the greek lemma of the entity from the knowledge base file

greekAlias1...greekAliasn: the greek alias row(s) of the entity from the KB



ID	Name	Type	Description	Surname	Alias1	Greek	GreekSurname	GreekAlias
168	Mary	Person		Magdalene		μαρία	μαγδαλην	
162	Caesarea	Place	Peter's confession	Philippi		Καισαρίας	Φιλίππου	

Figure 5: A graphical depiction of the AuthToken objects and a portion of the knowledge base schema

An object is created for each entity in the knowledge base and stored in the array for future processing.

Creating Candidate Objects

Once the knowledge base is converted into objects, it is time to take the list of text tokens and compare them with the knowledge base objects. If there is a match, then the process creates a new textToken object for each matched candidate. The entityMatch() function will loop through each token in the TextList array. If the text is in English, the code will check to see if the token is capitalized or not. The reason for this is that in the English language, proper nouns are typically formed by using a capital letter as the first character. However, there are instances where all characters are capitalized, for instance, in titles, headings, or when used for dramatic effect. One could go the route and just capitalize all tokens and knowledge base entities, then match. However, there are two reasons for using the check method: 1) performance-- only capitalize the words if there is a need, and; 2) converting a lowercase non-proper noun to all capitals in English may allow for false positive match. For instance, rose will not match the candidate Rose. However, converting to capital letters will produce ROSE, which will match the capital ROSE. Therefore, only when the word is all capital letters, like JOHN, in the text will the knowledge base object be converted to JOHN.

Once the determination is made on capitalization, the logic then turns to the following: if the text token in the list matches the knowledge base object's "name," "surname" or "alias_n" attributes, then append the knowledge base object's ID to a new list called candidateList. This list means that the text token has n number of possible candidates that the process must check. As we will see in the ambiguous() function in step four, if the length of this list is "1" meaning that only one possible match is found, then it is unambiguous. However, if the length of this list is greater than one, then it is an ambiguous candidate.

If the language is greek, the same process from above is used. However, instead of the knowledge base object's "name," "surname," and "alias_n" attributes, the greekLemma, greekSurname, and greekAlias_n are used. As with the English, these are appended to the candidateList. This logic for English and for Greek are the last component of the code that uses the text of the token or knowledge base until the final project is created. From this point forward, the ID values alone are used. Therefore, the first three steps are the only places where one need worry about the intricacies of each language. At this point, the objects are created for each text token with the following attributes:

TextToken object

ID: the number of the text token in order from $\text{word}_1 \ldots \text{word}_n$

tokenText: the actual text value of the token

assigned: boolean value indicated that there is a match

assignedAuthID: the assigned authority file ID of the matched candidate

pos: for future research on disambiguation of pronouns

ambiguous: boolean value indicating ambiguity

assignType: indicates which function determined the match

type: person/place

score: what was the winning candidate's Bayesian probability

ambCandidates: the list of possible candidates created in TextList above

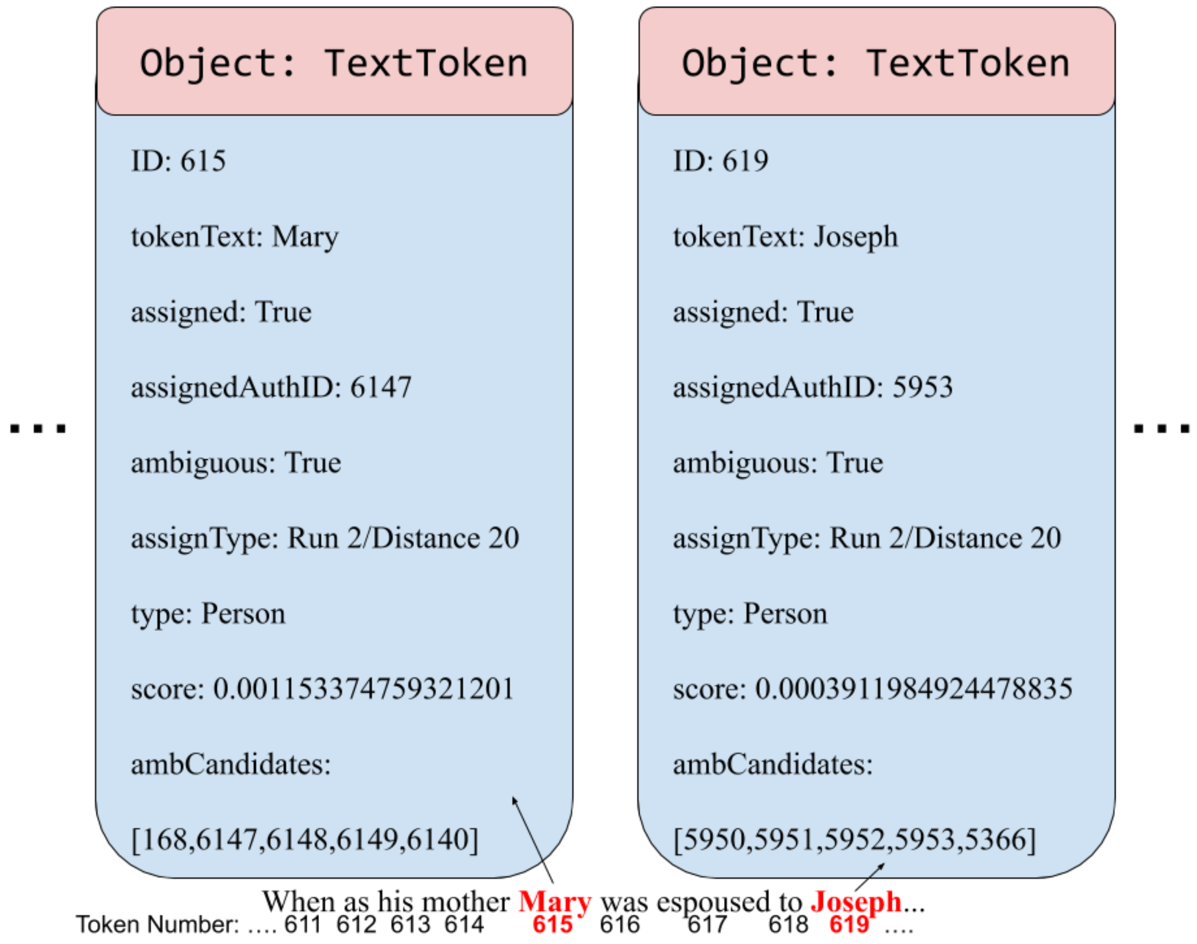


Figure 6: A graphic representing the TextTokens objects for tokens 615 and 619 at completion

As with the knowledge base entities, these are stored in an array of objects that we can loop through to get the various attribute information. At this point, we have two critical arrays: 1) and array of AuthToken objects containing our knowledge base information, and; 2) an array of TextToken objects containing text token information and attributes that we will update through future processes.

Determining Ambiguity and Matching Candidates

Once the information is processed using the aforementioned steps, determining ambiguity is quite straightforward. With three lines of code --one for loop and one if statement -- each textToken object's ambiguous attribute is updated with either True or False. For each object, if the length of the attribute ambCandidates is greater than one, then ambiguous is True; else ambiguous is false. From this point, we now have two paths for candidates: 1) unambiguous calculation and assignment, and; 2) ambiguous calculation and assignment.

Step Four: Calculate Unambiguous Candidates

Once we have all of our textToken objects with boolean values for the ambiguous attribute, the process is ready to calculate candidates that are unambiguous, or certain. In the current process, it is necessary to calculate and assign values to these candidates first because the process only uses candidates where assigned is True for scoring in the ambiguous calculations. Therefore, assigning the unambiguous or certain candidates is a crucial component in disambiguation. The calculateCertainCooccurence() function first loops through all textToken objects. As mentioned previously, if the object's ambCandidates attribute is equal to 1, then we know there is only one possible match found in the knowledge base. Therefore, given our input knowledge, we can match that candidate with certainty. The function updates the following attributes accordingly:

assigned = True

assignedAuthID = ambCandidates[0] (the only ambiguous candidate in the array) assignType = “Certain”

score = 1 (score ranges from 0 to 1)

The function then performs our first interaction with the tensor values. Since this is a match, it is also a connection that needs to be loaded into the tensor. Since the score of this match is a 1 in our probability (ranging from 0 to 1), we will update the tensor for the value itself.

a[*assignedAuthID, assignedAuthID, Text Number*] += 1

At this point, the process has determined that this token is an unambiguous token and it is assigned as such. However, the process then wants to know if there are any connections to this token found in the text? For this, the process uses cooccurrence. Co-occurrence is a variable set by the the program taking the average sentence length and dividing by 2.

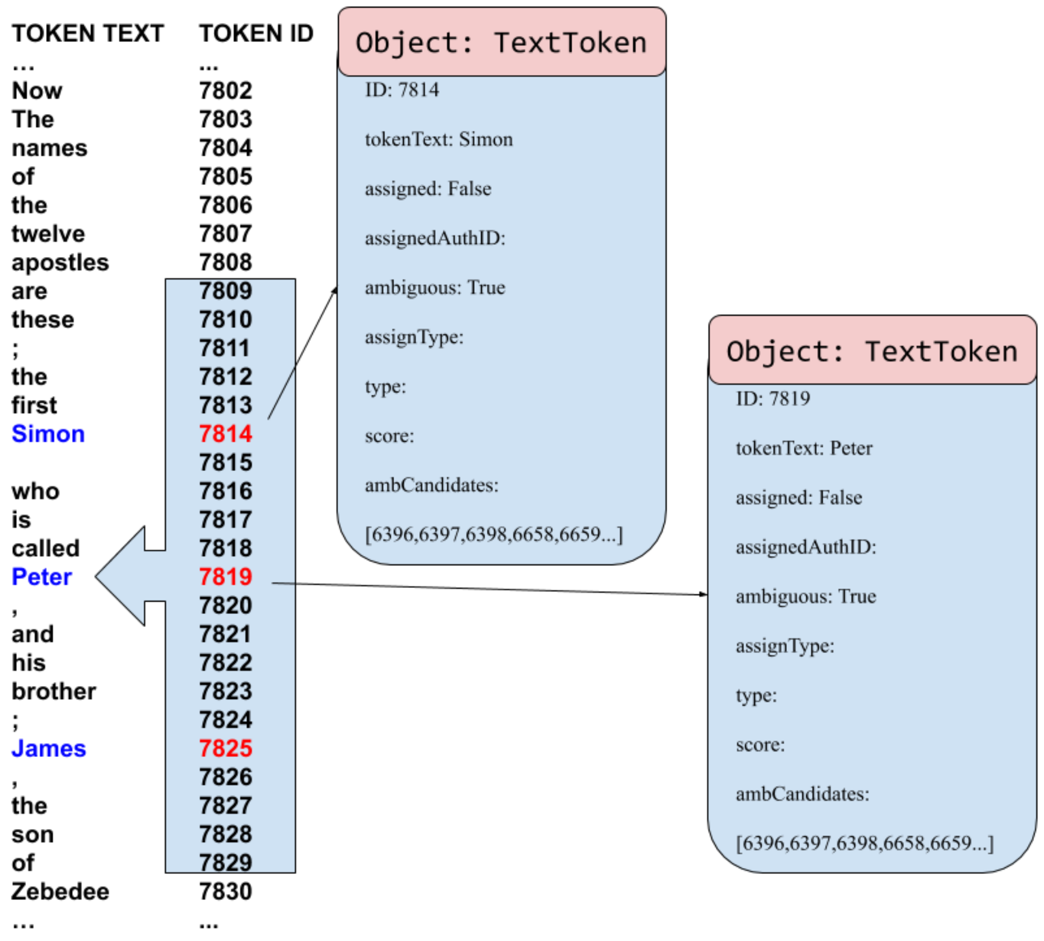


Figure 7: A depiction of a co-occurrence value of 10 and two corresponding TextToken objects

Therefore, for each of these certain tokens, say token 100, we loop back through the text and stop at each token that is within 10 places from 100. Therefore, tokens 90 through 110, not including 100, will be analyzed. If any token within this co-occurrence distance has an attribute of assigned = True (meaning that it is also a certain match), then this is a connection and the tensor is updated accordingly with a score of 1. For example, if obj.ID = 95 and obj.assigned = True, then the following occurs:

$$a[\text{ obj.ID}=95, \text{obj.ID}=100, \text{TextNumber}] \mathrel{+}= 1$$

and

a[obj.ID=100, obj.ID=95, TextNumber] += 1

At this point, the process has not determined all unambiguous candidates found in the raw text and updated the tensor with these new connections, thereby growing the knowledge available for any ambiguous calculation to come. The function returns the tensor a for future calculations.

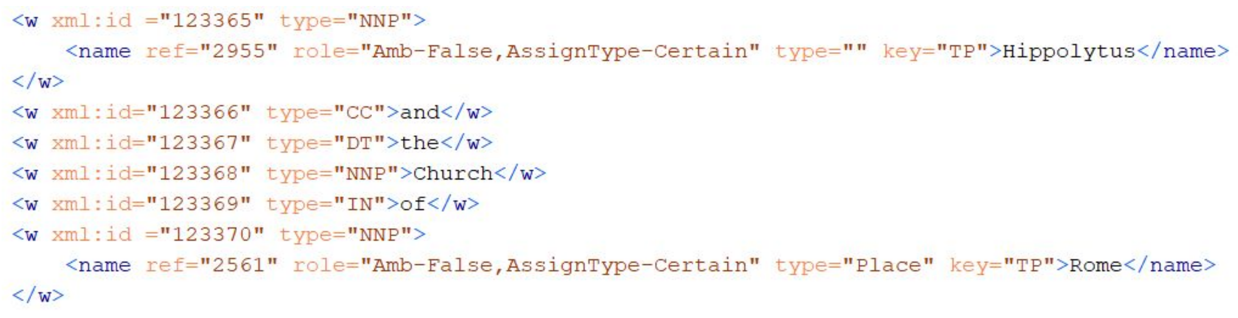
Calculate Certain Tokens

Text	Total Entities	# Unambiguous	# Ambiguous
Matthew- English	1,127	723 (64.15%)	404 (35.83%)
Matthew- Greek	900	814 (90.44%)	86 (9.55%)

Table 8: Number of ambiguous tokens in training datasets

Text	Function	Tagged Entities	Percentage
Matthew- English	calculateCertain()	710	0.62999
Matthew- Greek	calculateCertain()	802	0.89111

Table 9: Percentage from Matthew (English and Greek) training run after certain method



```
<w xml:id ="123365" type="NNP">
    <name ref="2955" role="Amb-False,AssignType-Certain" type="" key="TP">Hippolytus</name>
</w>
<w xml:id="123366" type="CC">and</w>
<w xml:id="123367" type="DT">the</w>
<w xml:id="123368" type="NNP">Church</w>
<w xml:id="123369" type="IN">of</w>
<w xml:id ="123370" type="NNP">
    <name ref="2561" role="Amb-False,AssignType-Certain" type="Place" key="TP">Rome</name>
</w>
```

Figure 8: Example of a tagged text for “Certain” (Unambiguous) tokens

Named Entity Disambiguation Process

Step Five: Matching Ambiguous Candidates Using only Surnames

The next step is to begin determining which ambiguous candidates are indeed the true candidate for the matched text token. At this point, the name, alias, and surname all indicate that the text token could possibly be more than one candidate. Therefore, we must make sure that a name + surname combination is unique among all candidates. To do this, the process calls the calculateSurname() function. This function loops through each object in TextTokens. If the assigned attribute is False, it then loops through each object in AuthTokens.

For the English language, a quick check is made to see if the TextToken.tokenText is equal to the AuthToken.name. If this is true, then we have a possible match. The process then loops through all possible candidates found in TextToken.ambCandidates. If that ambiguous candidate has a name and a surname that is found within +/- 3 tokens, then it is considered a match and the following attributes are updated:

assignedAuthID = ambCandidates[i]

assigned = True

assignType = “Surname”

score = 1

The same attribute values are assigned for the ambiguous candidate whose surname matched. At that point, the tensor is updated with a score of 1 for the matched value as follows:

a[*obj.ID, obj.ID, Text Number*] += 1

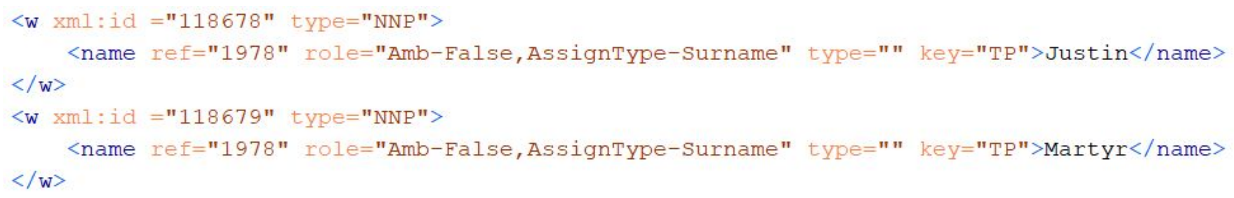
The process for the greek language is the same, except that it uses the attributes of greekLemma and greekSurname.

Following this assignment, the process then loops back through the text looking for any TextToken objects where assigned equals True and are within the co-occurrence distance. If there are, then these are connections that need to be updated in tensor a as follows:

a[*obj.ID, coocurrence.ID, Text Number]* += 1

Text	Function	Tagged Entities	Percentage
Matthew- English	calculateCertain()	710	0.62999
	+Surname	51	0.04525
Matthew- Greek	calculateCertain()	802	0.89111
	+Surname	22	0.02444

Table 10: Training results after Surname calculation



```
<w xml:id ="118678" type="NNP">
    <name ref="1978" role="Amb-False,AssignType-Surname" type="" key="TP">Justin</name>
</w>
<w xml:id ="118679" type="NNP">
    <name ref="1978" role="Amb-False,AssignType-Surname" type="" key="TP">Martyr</name>
</w>
```

Figure 9: Example tagged text using Surname logic

Step Six: Calculating Ambiguous

Scoring Candidates

The next function is where one sees the power of the tensor based algorithm for named entity disambiguation: the calculateAmbiguous() function. The function itself will call functions found in the BayesScore class to determine the probability that a candidate is the correct match.

At this point, we have matched certain tokens and tokens that have a unique corresponding surname. At this point, we are ready to begin the calculations to determine based off of name and $alias_n$.

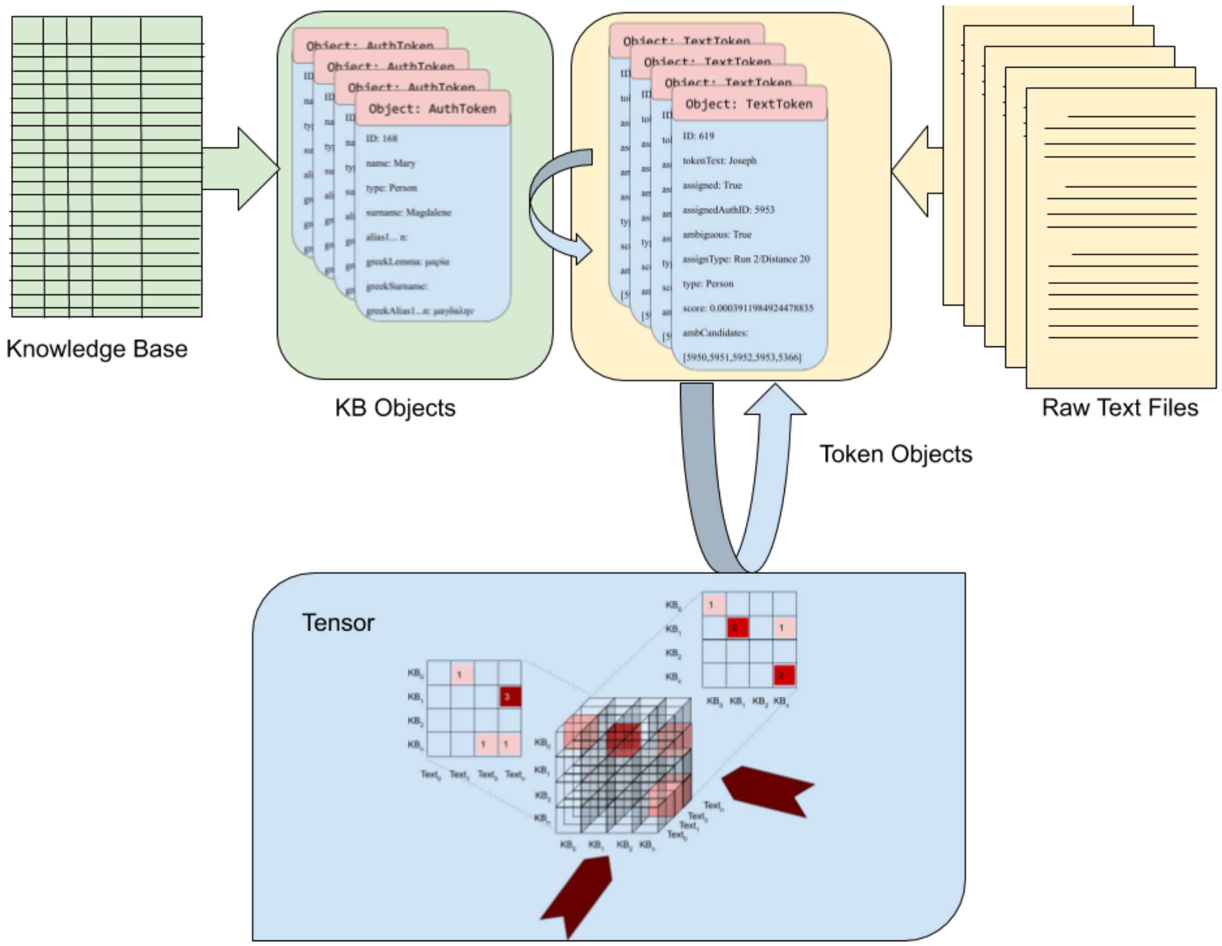


Figure 10: Graphic depiction of converting raw data into objects and the tensor for disambiguation

To begin, the process loops through each TextToken. If the ambiguous attribute is still True and assigned is False after the above methods, then we have an ambiguous candidate that still needs a match. At this point, the process calls on the Bayesian probability calculations [51]

$$P(A|B) = \frac{P(A) * P(B|A)}{P(B)}$$

Where P(A) is the probability that the candidate A will occur in the text and is calculated using the function called calculateP_A(). This is determined by looping through the tensor and summing the values for the following:

$$A = a[\ A.ID,\ y_0 \ldots y_n,\ text_0 \ldots text_n]$$

$$totalTokens = a[x_0 \ldots x_n,\ y_0 \ldots y_n,\ text_0 \ldots\ text_n]$$

$$P(A) = A/totalTokens$$

P(B) is the probability that all of the values within the co-occurrence will occur in the text and is calculated using the function called calculateP_B(). This is determined by looping through the tensor and summing values for the following:

$$B = [B.ID_0 \ldots B.ID_n,\ y_0 \ldots y_n,\ text_0 \ldots text_n]$$

$$P(B) = B/totalTokens$$

P(B|A) is the probability that all of the values within the co-occurence will occur in the text given that A is with them and is calculated using the function calculateP_B_A. This is determined by looping through the following:

$$P(B|A) = a[A.ID,\ B.ID_0 \ldots B.ID_n,\ text_0 \ldots text_n]/totalTokens$$

Given these values, the P(A|B) for each candidate is evaluated, which is the probability that A will occur given that B occurs in the text, and the candidate with the maximum P(A|B) score is assigned to the variable maxP_A_B along with the maximum ID number called maxPosition. The winning object's attributes are updated as follows:

assignedAuthID = maxPostion

```
        assigned = True
        score = maxP_A_B
        assignType = "Run:"+ str(run) + "w/ Amb Calc w/in =" + str(vardistance)
```

Learning on-the-fly and through Co-occurrence Window Growth and Multiple Iterations

When determining which entities to use in the Bayesian calculations above, testing determined that an objective measurement is to find the average sentence token length in the text and divide it by two. This provides the algorithm the ability to look for entities that are very close to the candidate on the first run. If there is a connection within this coccurence window, then the Bayesian probability calculations are used. If it is the first run and there is no match, then the candidate remains unassigned until subsequent runs. Any run after Run 1 will expand the window, as mentioned below.

Coccurence Window = Average Sentence Length / 2

Since the algorithm looks for entity connections before the candidate token and after the candidate token, the average sentence length is divided by 2 to that the window before the candidate token is half of the average sentence length and the window after the candidate token is half of the average sentence length. Therefore, the entire window for connections is equal to the average sentence length. The decision to use the average sentence length instead of relying on the user to input a window size is to use an objective method for determining the window length instead of a subjective user length.

The assignType variable is updated to reflect which iteration and at what co-occurrence level was this candidate assigned. This is because the algorithm, like humans, learns as new information is absorbed. As one saw during the previous functions, the algorithm is updating the tensor with new connections as it loops through the text tokens. With each new connection added to previous supervised and machine updated texts, subsequent tokens can use that information to make the best informed decision. On top of this, the algorithm also includes the ability to iterate through the text multiple times. For example, token.ID 40 may not have had any matches within 20 tokens on run 1. However, token.ID 50 may have been able to determine that it matched. Therefore, run 2 would allow token.ID 40 to use the connection with tokenID 50 to make a match. Also, after two distinct runs (which is run 3) the algorithm replicates what a human would do and expand the window that one can form the connection until it finds a suitable match. Therefore, if token.ID 200 goes through runs 1 and 2 with no candidates within the set co-occurrence limit (in our case, 20), then on run three it will start growing in size by 20 until a candidate with a connection is found. However, there should be a limit to this growth. Without a limit, one loses the benefit of using a Bayesian scoring system. With no limit, then one essentially just finds the probability that a candidate will occur in the text. So, candidate A = Joseph, husband of Mary and Candidate B = Joseph of Arimathaea would essentially just give you which Joseph is more likely to occur in this text, which in our case would be Candidate A. Yet, by growing it slowly until a candidate is found, then it allows us to use the Bayesian calculation to determine the probability that the candidate will occur, given the fact that candidate C occurs in the text. The algorithm therefore sets a growth limit of 1/5th the number of tokens in the test. So, if there are 50,000 tokens, the algorithm will expand the co-occurrence until it equals 1,000. This differs from the other algorithms due to the fact that many use TF/IDF, etc [24, 25].

Once the object with the highest probability is assigned, it is time to update the tensor with the new assignment and any connections that fall within the co-occurrence window. The tensor is updated with the following:

a[*MatchedObject.ID, CooccurenceObject.ID, Text Number*] += 1

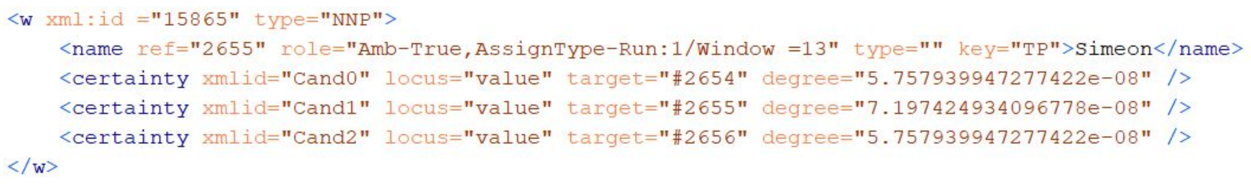
a[*CooccurenceObject.ID, MatchedObject.ID, Text Number*] += 1

a[*MatchedObject.ID, MatchedObject.ID, Text Number*] += 1

This makes the connection between each object and makes sure that the object itself is updated.

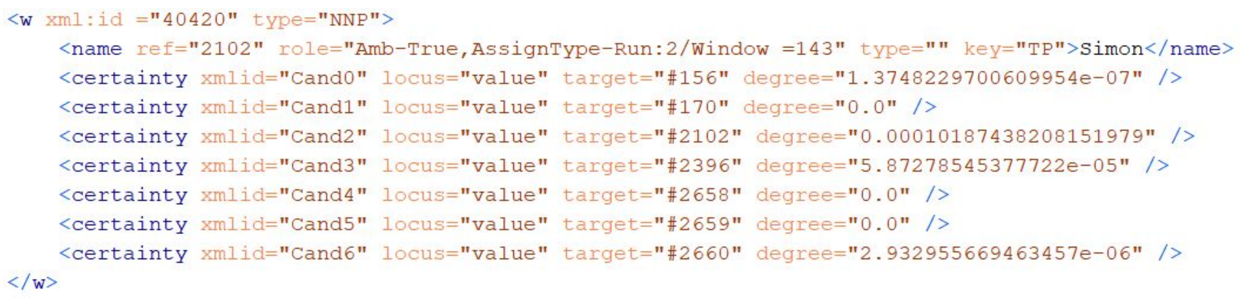
Text	**Function**	**Tagged Entities**	**Percentage**
Matthew- English	calculateCertain()	710	0.62999
	+Surname	51	0.04525
	+Run 1 (Static Window)	144	0.12777
	+Run 2 (Expanding Window)	222	0.19698
	+Run 3 (Expanding Window)	0	0.00000
Matthew- Greek	calculateCertain()	802	0.89111
	+Surname	22	0.02444
	+Run 1 (Static Window)	55	0.06111
	+Run 2 (Expanding Window)	21	0.02333
	+Run 3 (Expanding Window)	0	0.00000

Table 11: Results from training data after each calculation



```xml
<w xml:id ="15865" type="NNP">
    <name ref="2655" role="Amb-True,AssignType-Run:1/Window =13" type="" key="TP">Simeon</name>
    <certainty xmlid="Cand0" locus="value" target="#2654" degree="5.757939947277422e-08" />
    <certainty xmlid="Cand1" locus="value" target="#2655" degree="7.197424934096778e-08" />
    <certainty xmlid="Cand2" locus="value" target="#2656" degree="5.757939947277422e-08" />
</w>
```

Figure 11: Example of a tagged text matched and score by the ambiguous calculation method



```xml
<w xml:id ="40420" type="NNP">
    <name ref="2102" role="Amb-True,AssignType-Run:2/Window =143" type="" key="TP">Simon</name>
    <certainty xmlid="Cand0" locus="value" target="#156" degree="1.3748229700609954e-07" />
    <certainty xmlid="Cand1" locus="value" target="#170" degree="0.0" />
    <certainty xmlid="Cand2" locus="value" target="#2102" degree="0.00010187438208151979" />
    <certainty xmlid="Cand3" locus="value" target="#2396" degree="5.87278545377722e-05" />
    <certainty xmlid="Cand4" locus="value" target="#2658" degree="0.0" />
    <certainty xmlid="Cand5" locus="value" target="#2659" degree="0.0" />
    <certainty xmlid="Cand6" locus="value" target="#2660" degree="2.932955669463457e-06" />
</w>
```

Figure 12: Example of tagged text using an expanded co-occurrence window

Additional Components

Part-of-Speech Tagging

For each token, the process using the Natural Language Toolkit (NLTK) library to add additional "Part-of-Speech" tags to each token. While this process may not be specifically needed for our current ambiguous candidate process, this information is beneficial in two ways: 1) for future research on finding unknown entities in an unsupervised fashion, and; 2) to provide future end users analytical linguistic capabilities for queries. Therefore, it made sense to proceed with coding this functionality into the process now for the benefit of future research opportunities and current academic scholarship analysis of the texts.

Printing and Tagging in TEI

After the aforementioned functions run through all iterations and growth, we are now able to output a file of tagged text entities. The goal of this project was to demonstrate: 1) the ease of creating a simple knowledge base, 2) the feasibility of using a tensor based approach to disambiguation, and; 3) creating a multilingual digital humanities tool for use on early Christian texts from late-antiquity. The end result of this are XML files for each text encoded in TEI standard that one can load into a no-sql database for analysis.

For the purposes of this research, it is important to see specific information for each candidate that in the academic use of the tool will not need access. For instance, this research needs to see all candidates, the scores, which method assigned the match, etc.

Recommending Possible Candidates

Committee member Dr. Anderson provided a question during the dissertation proposal regarding entities in the text that are not found in the knowledge base. Implementing this concept fully -- where the program uploads the unknown candidates into the tensor in an unsupervised fashion -- will take further research and development. At this time, it does not appear that the natural language processing tools are sufficiently accurate with this genre to find only unknown proper nouns. However, this project does give the user the added functionality to run the findAuthorityCandidates() function. This function uses the NLTK library to tokenize and return all possible proper nouns. From there, the program loops through and compares each proper noun token with each entity (name, surname, alias, etc) found in the knowledge base file. If there is not a match, it is added to a comma delimited file that the user can look at to determine if the possible candidate is indeed an unknown entity. While this feature is not an unsupervised approach to finding unknown candidates, it is a vast improvement over having to manually comb through each text looking for untagged entities. However, at this time, the NLP functionality available using the NLTK package was not accurate enough to upload the non-matched unknowns into the tensor without human supervision.

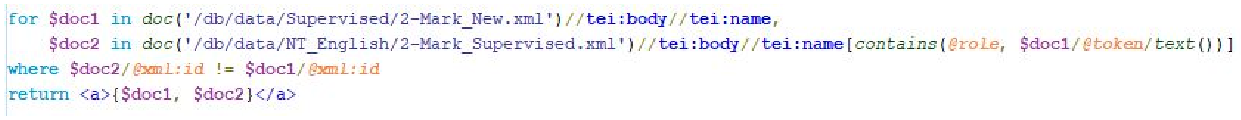
There are two new possibilities available for future research that may solve the accuracy of the NLTK package: BERT and ELMo from AllenNLP. Devlin et al just published a journal article titled *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* BERT is a pre-trained model whose "...aim [is] to predict the relationships between sentences by analyzing them holistically, as well as token-level tasks such as named entity recognition and question answering" [56]. The paper states that there are two strategies for these tasks: feature-based and fine-tuning. BERT falls into the latter category. The authors propose the following in the article, "BERT is the first finetuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many task-specific architectures" while providing a further claim that "BERT advances the state of the art for eleven NLP tasks" [56]. Given these claims, one should consider this method for further research in finding unknown entities in raw text as well as being used for the tokenization process.

The second new possibility is mentioned by Devlin et al as being part of the "feature-based" strategies, ELMo by AllenNLP [56]. AllenNLP uses a model "pre-trained on a large corpus" [57]. While the use for Named Entity Extraction may not be suitable for domain specific tasks such as our research, the NLP functionality used for tokenizing and part-of-speech tagging looks very promising. While this falls into the "Future Research" category, early testing for use in the current algorithm indicate that the part-of-speech tagging found in the ELMo may very well find proper nouns with greater accuracy than the current NLTK library in Python.

Results

The Benefits of TEI Documents

Since the product of the tool are XML documents encoded in the TEI standard, one only need to upload the documents into a no-sql database such as eXist-DB and compare with a file containing supervised tags for all entities. Using XQuery, one can use the following query to determine if the assigned entity ID for a given token matches the entity ID in the supervised file.



```
for $doc1 in doc('/db/data/Supervised/2-Mark_New.xml')//tei:body//tei:name,
    $doc2 in doc('/db/data/NT_English/2-Mark_Supervised.xml')//tei:body//tei:name[contains(@role, $doc1/@token/text())]
where $doc2/@xml:id != $doc1/@xml:id
return <a>{$doc1, $doc2}</a>
```

Figure 13: A query in the XQuery language to return mismatched IDs

Matthew Training

As mentioned previously, the training data for this research consists of the four gospels -- Matthew, Mark, Luke, and John -- found in the Christian New Testament. The goal is to use these four texts which depict a similar narrative, but in different ways, to train the algorithm and test the various methods during the software development phase. The thesis is that if the computer can use a knowledge base for the four gospel books to accurately predict ambiguous candidates, then the same method can be used on any other corpora of texts, such as the Ante-Nicene writings in our current research.

The following are the results from the training runs during the development phase of the research:

Training Scoring

Text	Precision	Recall	F-Score
Matthew- English	0.99470	0.99383	0.99427
Matthew- Greek	0.97614	0.96774	0.97192

Table 12: Scoring results on each training text for Matthew (English and Greek)

The algorithm produced excellent results during the training for software development. A note of caution: the knowledge base for these four texts was created directly from the texts and after numerous iterations of trial and error. For instance, the research would iterate through Matthew in an unsupervised fashion. When complete, a supervised step would manually look at the results and find any issues. Depending on the issue, the supervised step would add data to the knowledge base -- for instance, add a new entity, a new connection, a new alias, etc -- or additional code and logic would be added to the algorithm to correct the issue. Therefore, one would expect a very high score on each since there would be very few false negatives due to unknown entities and the algorithm would be “fitted” almost exclusively for these texts.

However, as predicted, these texts provided an exceptional training set for developing the algorithm and for software development testing.

Testing Scoring

Text	Precision	Recall	F-Score
5_Vol_1_1_1_Clement_of_Rome_Introduction	1.00000	0.86301	0.92647
6_Vol_1_1_Clement_of_Rome	0.97770	0.95811	0.96780
7_Vol_1_2_1_Mathetes_Introduction	0.90476	0.73077	0.80851
8_Vol_1_2_Mathetes	0.98980	0.91509	0.95098
9_Vol_1_3_a_1_Polycarp_Epistle_to_the_Philippians_Introduction	0.96970	0.88889	0.92754
10_Vol_1_3_a_Polycarp_Epistle_to_the_Philippians	1.00000	0.95105	0.97491
11_Vol_1_3_b_1_Polycarp_Martyrdom_Introduction	0.76190	0.94118	0.84211
12_Vol_1_3_b_Polycarp_Martyrdom	0.96552	0.93333	0.94915
13_Vol_1_4_1_Ignatius_Epistles_Introduction	0.94872	0.87402	0.90984
14_Vol_1_4_a_Ignatius_Epistle_to_the_Ephesians	0.98963	0.98554	0.98758
15_Vol_1_4_b_Ignatius_Epistle_to_the_Magnesians	1.00000	0.98392	0.99190
16_Vol_1_4_c_Ignatius_Epistle_to_the_Trallians	0.99310	0.97297	0.98294
17_Vol_1_4_d_Ignatius_Epistle_to_the_Philadelphians	0.99140	0.94536	0.96783
18_Vol_1_4_e_Ignatius_Epistle_to_the_Romans	0.97881	0.99569	0.98718
19_Vol_1_4_f_Ignatius_Epistle_to_the_Smyrnæans	0.99303	0.96610	0.97938
20_Vol_1_4_g_Ignatius_Epistle_to_the_Polycarp	0.97183	0.95172	0.96167
21_Vol_1_4_h_1_Ignatius_Syriac_Introduction	1.00000	0.75000	0.85714
22_Vol_1_4_h_Ignatius_Syriac	0.99408	0.98246	0.98824
23_Vol_1_4_i_1_Ignatius_Spurious_Epistles_Introduction	0.94444	0.73913	0.82927

24_Vol_1_4_i_Ignatius_Spurious_Epistles	0.99104	0.96232	0.97647
25_Vol_1_4_j_1_Ignatius_Martyrdom_Introduction	0.95238	0.83333	0.88889
26_Vol_1_4_j_Ignatius_Martyrdom	0.97500	0.91406	0.94355
27_Vol_1_5_1_Barnabas_Introduction	0.93023	0.75472	0.83333
28_Vol_1_5_Barnabas	0.98214	0.97778	0.97996
29_Vol_1_6_1_Papias_Introduction	1.00000	0.97059	0.98507
30ol_1_6_Papias	0.93296	0.91758	0.92521
31_Vol_1_7_1_Justin_Martyr_Introduction	0.96250	0.79381	0.87006
32_Vol_1_7_a_Justin_Martyr_First_Apology	0.97778	0.93015	0.95337
33_Vol_1_7_b_Justin_Martyr_Second_Apology	0.91549	0.90278	0.90909
34_Vol_1_7_c_Justin_Martyr_Dialouge_with_Trypho	0.99020	0.95946	0.97459
35_Vol_1_7_d_Justin_Martyr_Discourse_to_the_Greeks	1.00000	0.65979	0.79503
36_Vol_1_7_e_Justin_Martyr_Hortatory_Address_to_the_Greeks	0.99363	0.86069	0.92239
37_Vol_1_7_f_Justin_Martyr_On_the_Sole_Government_of_God	0.98113	0.85950	0.91630
38_Vol_1_7_g_Justin_Martyr_On_the_Resurrection	1.00000	0.98039	0.99010
39_Vol_1_7_h_Justin_Martyr_Other_Fragments	0.97674	0.98824	0.98246
40_Vol_1_7_i_Justin_Martyr_Martydom	0.88095	0.89516	0.88800
41_Vol_1_8_a_1_Irenaeus_Against_Heresies_Introduction	0.93617	0.62411	0.74894
42_Vol_1_8_a_Irenaeus_Against_Heresies	0.98763	0.91860	0.95186
43_Vol_1_8_b_Irenaeus_Fragments	0.99555	0.88166	0.93515
TOTAL Volume One	**0.98535**	**0.92737**	**0.95548**

Table 13: Measurements for Precision, Recall, and F-score on Test Data

Measurement Analysis

Table 13 provides a look at the results of the program on volume one of the Ante-Nicene Fathers. The process scanned 645,579 tokens without the supervision of a human in the tagging process. Of these 645,579 tokens, there were 22,391 entity candidates found by the computer, or 3.47% of the tokens were tagged candidates. Table 13 provides three different scores for analyzing the results. The first score is known as Precision. Precision is a measurement of how many…

$$\text{Precision} = |tp| / |tp| + |fp|$$

The second score is known as Recall. Recall is a measurement of how many…

$$\text{Recall} = |tp| / |tp| + |fn|$$

And the final score is known as an F-score. The F-score is…

$$F1 = 2 \text{ x Precision x Recall / Precision + Recall}$$

[28].

Given these measurements, one can see that the overall F-score for the texts is 0.95548. When analyzing these measurements, one sees that Precision is very high, with an average of over .98. This means that the algorithm did well finding candidates that were known to the knowledge base; that is, if the token exists in the knowledge base, then it tagged the token. Given the fact that this research uses indices or dictionaries for the knowledge base and does not attempt to find unknown candidates in an unsupervised fashion, this is an excellent measurement.

The factor bringing down the F-score to a little over .95 is the Recall score. In this research, the Recall is much lower than the precision due to the fact that there were many entities that were not found in the General Index for the volumes. For instance, in text number 42, Irenaeus's *Against Heresies,* one finds 932 False Negatives; this took the F-score down to just above a .95. However, when analyzing the false negatives, only Y of these entities were known to the knowledge base. This means, that the algorithm only missed Y candidates. Using this data and assuming that the algorithm would have a precision of at least .98 if these were in knowledge base beforehand, we predict that the F-score would be considerably higher, probably around .98.

At this time, the natural language toolkits are not sophisticated enough to rely on finding proper nouns for unsupervised additions to the knowledge base. This concept was tested during this research, but found that too many non-proper nouns were tagged and introduced too much error. However, as noted below in the future research section, using natural language processing to identify unknown entities in an unsupervised fashion is ripe for future research.

Another note regarding the final results: there seems to be a noticable difference when analyzing editor introductions from the primary source material. In each case, the editor introductory notes -- which were tagged using the algorithm -- are considerably lower F-score than the actual primary source material. For the reasons noted above, it appears that the editors used many entities in their introductory notes that were not used in the General Index. Therefore, since these entities were not found in the index, they did not make it into the knowledge base for use in the algorithm. However, removing the introductory notes improved the F-score by only .003 points due to the fact that introductions are a small percentage of the total tokens in the volume.

Certain Match Analysis and Examples

Text	Certain Matches	Percent

5_Vol_1_1_1_Clement_of_Rome_Introduction	36	0.57143
6_Vol_1_1_Clement_of_Rome	324	0.60223
7_Vol_1_2_1_Mathetes_Introduction	20	0.95238
8_Vol_1_2_Mathetes	63	0.64286
9_Vol_1_3_a_1_Polycarp_Epistle_to_the_Philippians_Introduction	17	0.51515
10_Vol_1_3_a_Polycarp_Epistle_to_the_Philippians	88	0.64706
11_Vol_1_3_b_1_Polycarp_Martyrdom_Introduction	16	0.76190
12_Vol_1_3_b_Polycarp_Martyrdom	94	0.54023
13_Vol_1_4_1_Ignatius_Epistles_Introduction	99	0.84615
14_Vol_1_4_a_Ignatius_Epistle_to_the_Ephesians	259	0.53734
15_Vol_1_4_b_Ignatius_Epistle_to_the_Magnesians	173	0.56536
16_Vol_1_4_c_Ignatius_Epistle_to_the_Trallians	184	0.63448
17_Vol_1_4_d_Ignatius_Epistle_to_the_Philadelphians	201	0.57593
18_Vol_1_4_e_Ignatius_Epistle_to_the_Romans	120	0.50847
19_Vol_1_4_f_Ignatius_Epistle_to_the_Smyrnæans	163	0.56794
20_Vol_1_4_g_Ignatius_Epistle_to_the_Polycarp	65	0.45775
21_Vol_1_4_h_1_Ignatius_Syriac_Introduction	16	0.88889
22_Vol_1_4_h_Ignatius_Syriac	95	0.56213
23_Vol_1_4_i_1_Ignatius_Spurious_Epistles_Introduction	15	0.83333
24_Vol_1_4_i_Ignatius_Spurious_Epistles	432	0.64478
25_Vol_1_4_j_1_Ignatius_Martyrdom_Introduction	39	0.92857
26_Vol_1_4_j_Ignatius_Martyrdom	98	0.81667
27_Vol_1_5_1_Barnabas_Introduction	37	0.86047
28_Vol_1_5_Barnabas	301	0.67188
29_Vol_1_6_1_Papias_Introduction	27	0.81818

30ol_1_6_Papias	109	0.60894
31_Vol_1_7_1_Justin_Martyr_Introduction	58	0.72500
32_Vol_1_7_a_Justin_Martyr_First_Apology	671	0.64831
33_Vol_1_7_b_Justin_Martyr_Second_Apology	104	0.73239
34_Vol_1_7_c_Justin_Martyr_Dialouge_with_Trypho	2330	0.61706
35_Vol_1_7_d_Justin_Martyr_Discourse_to_the_Greeks	52	0.81250
36_Vol_1_7_e_Justin_Martyr_Hortatory_Address_to_the_Greeks	336	0.53503
37_Vol_1_7_f_Justin_Martyr_On_the_Sole_Government_of_God	61	0.57547
38_Vol_1_7_g_Justin_Martyr_On_the_Resurrection	46	0.46000
39_Vol_1_7_h_Justin_Martyr_Other_Fragments	54	0.62791
40_Vol_1_7_i_Justin_Martyr_Martydom	110	0.87302
41_Vol_1_8_a_1_Irenaeus_Against_Heresies_Introduction	57	0.60638
42_Vol_1_8_a_Irenaeus_Against_Heresies	6702	0.62800
43_Vol_1_8_b_Irenaeus_Fragments	267	0.59465
TOTAL	**13939**	**0.62233**

Table 14: Certain (Unambiguous) Match analysis on test data

The first method in the ambiguous candidate matching determines if the candidate is a certain match, or unambiguous. This is a simple calculation where the name, surname, and all alias fields are compared to the token. If there is only one possible match for the token, then the algorithm tags it as a certain match. This is a very important step in the algorithm and for our analysis for three reasons: 1) a certain match means that we found a match for the token candidate; 2) a certain match is necessary to form the base connections for the ambiguous matches. Without these certain matches, there would be no neighbors to form connections in the subsequent processes, and; 3) the certain matches give us an indication of what percentage of the text is unambiguous versus ambiguous. In our test case, around 62% of the candidate tokens were unambiguous. Therefore, this is our baseline for what a computer can find without the need for named entity disambiguation. The rest of the process is to see if the computer can determine the other 38% of the ambiguous tokens.

Surname Analysis and Examples

Text	**Surname Match**	**Percent**
5_Vol_1_1_1_Clement_of_Rome_Introduction	0	0.00000
6_Vol_1_1_Clement_of_Rome	16	0.02974
7_Vol_1_2_1_Mathetes_Introduction	0	0.00000
8_Vol_1_2_Mathetes	2	0.02041
9_Vol_1_3_a_1_Polycarp_Epistle_to_the_Philippians_Introduction	0	0.00000
10_Vol_1_3_a_Polycarp_Epistle_to_the_Philippians	0	0.00000
11_Vol_1_3_b_1_Polycarp_Martyrdom_Introduction	0	0.00000
12_Vol_1_3_b_Polycarp_Martyrdom	9	0.05172
13_Vol_1_4_1_Ignatius_Epistles_Introduction	1	0.00855
14_Vol_1_4_a_Ignatius_Epistle_to_the_Ephesians	12	0.02490
15_Vol_1_4_b_Ignatius_Epistle_to_the_Magnesians	6	0.01961
16_Vol_1_4_c_Ignatius_Epistle_to_the_Trallians	4	0.01379
17_Vol_1_4_d_Ignatius_Epistle_to_the_Philadelphians	6	0.01719
18_Vol_1_4_e_Ignatius_Epistle_to_the_Romans	2	0.00847
19_Vol_1_4_f_Ignatius_Epistle_to_the_Smyrnæans	7	0.02439
20_Vol_1_4_g_Ignatius_Epistle_to_the_Polycarp	0	0.00000

21_Vol_1_4_h_1_Ignatius_Syriac_Introduction	0	0.00000
22_Vol_1_4_h_Ignatius_Syriac	2	0.01183
23_Vol_1_4_i_1_Ignatius_Spurious_Epistles_Introduction	2	0.11111
24_Vol_1_4_i_Ignatius_Spurious_Epistles	12	0.01791
25_Vol_1_4_j_1_Ignatius_Martyrdom_Introduction	2	0.04762
26_Vol_1_4_j_Ignatius_Martyrdom	1	0.00833
27_Vol_1_5_1_Barnabas_Introduction	0	0.00000
28_Vol_1_5_Barnabas	0	0.00000
29_Vol_1_6_1_Papias_Introduction	0	0.00000
30ol_1_6_Papias	3	0.01676
31_Vol_1_7_1_Justin_Martyr_Introduction	14	0.17500
32_Vol_1_7_a_Justin_Martyr_First_Apology	22	0.02126
33_Vol_1_7_b_Justin_Martyr_Second_Apology	2	0.01408
34_Vol_1_7_c_Justin_Martyr_Dialouge_with_Trypho	47	0.01245
35_Vol_1_7_d_Justin_Martyr_Discourse_to_the_Greeks	0	0.00000
36_Vol_1_7_e_Justin_Martyr_Hortatory_Address_to_the_Greeks	6	0.00955
37_Vol_1_7_f_Justin_Martyr_On_the_Sole_Government_of_God	0	0.00000
38_Vol_1_7_g_Justin_Martyr_On_the_Resurrection	0	0.00000
39_Vol_1_7_h_Justin_Martyr_Other_Fragments	1	0.01163
40_Vol_1_7_i_Justin_Martyr_Martydom	0	0.00000
41_Vol_1_8_a_1_Irenaeus_Against_Heresies_Introduction	0	0.00000
42_Vol_1_8_a_Irenaeus_Against_Heresies	103	0.00965
43_Vol_1_8_b_Irenaeus_Fragments	12	0.02673
TOTAL	**294**	**0.01313**

Table 15: Surname analysis on test data

The first true named entity disambiguation step in the test process is to analyze any possible candidates for an included surname described above in the surname section of the algorithm. Here, we see that around 1.3% of the total candidate tokens -- this includes the certain tokens -- were matched based on a having a surname included in the text. This is a small amount, but when looking at almost 650,000 tokens, this produced 295 matched candidates. Since an entity with a surname gives a higher probability of being a correct match, these 1.3% of the candidate tokens means we have a high confidence in the match. Therefore, this step -- even though only a small percentage -- is an important step in increasing the possible matches for further ambiguous calculation methods.

Run 1: Ambiguous within Co-occurrence Window Only Analysis and Examples

Text	**Run 2 Match**	**Percent**
5_Vol_1_1_1_Clement_of_Rome_Introduction	4	0.06349
6_Vol_1_1_Clement_of_Rome	43	0.07993
7_Vol_1_2_1_Mathetes_Introduction	1	0.04762
8_Vol_1_2_Mathetes	7	0.07143
9_Vol_1_3_a_1_Polycarp_Epistle_to_the_Philippians_Introduction	0	0.00000
10_Vol_1_3_a_Polycarp_Epistle_to_the_Philippians	24	0.17647
11_Vol_1_3_b_1_Polycarp_Martyrdom_Introduction	1	0.04762
12_Vol_1_3_b_Polycarp_Martyrdom	22	0.12644
13_Vol_1_4_1_Ignatius_Epistles_Introduction	7	0.05983
14_Vol_1_4_a_Ignatius_Epistle_to_the_Ephesians	106	0.21992
15_Vol_1_4_b_Ignatius_Epistle_to_the_Magnesians	64	0.20915

16_Vol_1_4_c_Ignatius_Epistle_to_the_Trallians	63	0.21724
17_Vol_1_4_d_Ignatius_Epistle_to_the_Philadelphians	81	0.23209
18_Vol_1_4_e_Ignatius_Epistle_to_the_Romans	54	0.22881
19_Vol_1_4_f_Ignatius_Epistle_to_the_Smyrnæans	54	0.18815
20_Vol_1_4_g_Ignatius_Epistle_to_the_Polycarp	18	0.12676
21_Vol_1_4_h_1_Ignatius_Syriac_Introduction	0	0.00000
22_Vol_1_4_h_Ignatius_Syriac	31	0.18343
23_Vol_1_4_i_1_Ignatius_Spurious_Epistles_Introduction	1	0.05556
24_Vol_1_4_i_Ignatius_Spurious_Epistles	104	0.15522
25_Vol_1_4_j_1_Ignatius_Martyrdom_Introduction	0	0.00000
26_Vol_1_4_j_Ignatius_Martyrdom	7	0.05833
27_Vol_1_5_1_Barnabas_Introduction	2	0.04651
28_Vol_1_5_Barnabas	42	0.09375
29_Vol_1_6_1_Papias_Introduction	0	0.00000
30ol_1_6_Papias	19	0.10615
31_Vol_1_7_1_Justin_Martyr_Introduction	1	0.01250
32_Vol_1_7_a_Justin_Martyr_First_Apology	156	0.15072
33_Vol_1_7_b_Justin_Martyr_Second_Apology	14	0.09859
34_Vol_1_7_c_Justin_Martyr_Dialouge_with_Trypho	669	0.17717
35_Vol_1_7_d_Justin_Martyr_Discourse_to_the_Greeks	4	0.06250
36_Vol_1_7_e_Justin_Martyr_Hortatory_Address_to_the_Greeks	64	0.10191
37_Vol_1_7_f_Justin_Martyr_On_the_Sole_Government_of_God	5	0.04717
38_Vol_1_7_g_Justin_Martyr_On_the_Resurrection	9	0.09000
39_Vol_1_7_h_Justin_Martyr_Other_Fragments	16	0.18605
40_Vol_1_7_i_Justin_Martyr_Martydom	9	0.07143

41_Vol_1_8_a_1_Irenaeus_Against_Heresies_Introduction	7	0.07447
42_Vol_1_8_a_Irenaeus_Against_Heresies	1710	0.16023
43_Vol_1_8_b_Irenaeus_Fragments	58	0.12918
TOTAL Volume One	**3477**	**0.15524**

Table 16: Results of first-run Ambiguous method on test data

Up until this point, the computer has found all certain matches; that is, candidates with one and only one possible match. The computer has also ascertained all matches with included surnames in the text. The next step is to start finding matches for candidates with more than one possible entity match: ambiguous entities. Due to the fact that the algorithm uses co-occurrence to determine connections, this means that only tokens within the co-occurrence window that are already tagged (ie: certain and surname only) can be used during “Run 1”. Therefore, the process will need to run more than once because everytime we find a match, this increases the connections. On “Run 1” things are kept simple in order to make sure that there is a high likelihood on the match. The window stays static at average sentence length divided by two described above. This means that any connection is a close connection linguistically; that is, the connection tokens are very close to the candidate token.

Running this step adds another 15.5% to our matched token total. We also know that these tokens were matched based on connection tokens that were very close in proximity to the candidate token. Therefore, not only does this get us closer to 100% match rate, but it also adds another 3,477 matches -- and connections -- to use in our future runs. This keeps building the network for future named entity disambiguation calculations.

Run 2: Ambiguous Match with Co-occurrence and Growing Window Analysis

Text	Run 2 Matches	Percentage
5_Vol_1_1_1_Clement_of_Rome_Introduction	22	0.34921
6_Vol_1_1_Clement_of_Rome	152	0.04089
7_Vol_1_2_1_Mathetes_Introduction	0	0.00000
8_Vol_1_2_Mathetes	26	0.26531
9_Vol_1_3_a_1_Polycarp_Epistle_to_the_Philippians_Introduction	15	0.45455
10_Vol_1_3_a_Polycarp_Epistle_to_the_Philippians	24	0.17647
11_Vol_1_3_b_1_Polycarp_Martyrdom_Introduction	3	0.14286
12_Vol_1_3_b_Polycarp_Martyrdom	50	0.28736
13_Vol_1_4_1_Ignatius_Epistles_Introduction	13	0.11111
14_Vol_1_4_a_Ignatius_Epistle_to_the_Ephesians	105	0.21784
15_Vol_1_4_b_Ignatius_Epistle_to_the_Magnesians	63	0.20588
16_Vol_1_4_c_Ignatius_Epistle_to_the_Trallians	39	0.13448
17_Vol_1_4_d_Ignatius_Epistle_to_the_Philadelphians	61	0.17479
18_Vol_1_4_e_Ignatius_Epistle_to_the_Romans	60	0.25424
19_Vol_1_4_f_Ignatius_Epistle_to_the_Smyrnæans	63	0.21951
20_Vol_1_4_g_Ignatius_Epistle_to_the_Polycarp	57	0.40141
21_Vol_1_4_h_1_Ignatius_Syriac_Introduction	2	0.11111
22_Vol_1_4_h_Ignatius_Syriac	41	0.24260
23_Vol_1_4_i_1_Ignatius_Spurious_Epistles_Introduction	0	0.00000
24_Vol_1_4_i_Ignatius_Spurious_Epistles	122	0.18209
25_Vol_1_4_j_1_Ignatius_Martyrdom_Introduction	1	0.02381
26_Vol_1_4_j_Ignatius_Martyrdom	14	0.11667
27_Vol_1_5_1_Barnabas_Introduction	3	0.06977

28_Vol_1_5_Barnabas	105	0.23438
29_Vol_1_6_1_Papias_Introduction	5	0.15152
30ol_1_6_Papias	44	0.24581
31_Vol_1_7_1_Justin_Martyr_Introduction	6	0.07500
32_Vol_1_7_a_Justin_Martyr_First_Apology	182	0.17585
33_Vol_1_7_b_Justin_Martyr_Second_Apology	20	0.14085
34_Vol_1_7_c_Justin_Martyr_Dialouge_with_Trypho	730	0.19333
35_Vol_1_7_d_Justin_Martyr_Discourse_to_the_Greeks	7	0.10938
36_Vol_1_7_e_Justin_Martyr_Hortatory_Address_to_the_Greeks	218	0.34713
37_Vol_1_7_f_Justin_Martyr_On_the_Sole_Government_of_God	39	0.36792
38_Vol_1_7_g_Justin_Martyr_On_the_Resurrection	43	0.43000
39_Vol_1_7_h_Justin_Martyr_Other_Fragments	13	0.15116
40_Vol_1_7_i_Justin_Martyr_Martydom	7	0.05556
41_Vol_1_8_a_1_Irenaeus_Against_Heresies_Introduction	29	0.30851
42_Vol_1_8_a_Irenaeus_Against_Heresies	2139	0.20043
43_Vol_1_8_b_Irenaeus_Fragments	112	0.24944
TOTAL Volume One	**4635**	**0.20694**

Table 17: Second-run Ambiguous calculation with expanding co-occurrence window

The next step in the named entity disambiguation process is to run through another cycle of the ambiguous calculation method. Why would one want to run the same process again? Because this cycle now has the additional 3,477 tokens (or 15.5% more tokens) to use for connections to determine an appropriate match. Not only does “Run 2” have these additional connections, but this is also the step where the computer also expands the co-occurrence window until one of two things occur: 1) connections are found within the window width that allow for a match to take place, or; 2) the window size has grown to 1⁄5 of the total text tokens without a connection. In case one, a match is made, the text is tagged, and the tensor is updated. In case two, there is not enough information provided to make a determination. Therefore, since there is a chance that candidate tokens fall into the second case, we will need to perform a third run. Why would a third run possibly find a match when run 1 and 2 did not? Because “Run 2” produced 4.635 new matches adding over 20% of our total matches to the tensor. Therefore, it is possible that a third run -- now with access to these 4,635 connections -- can find some that run two did not.

Run 3: Rerun Ambiguous Match with Co-occurrence and Growing Window Analysis and Examples

Text	**Run 3 Match**	**Percent**
5_Vol_1_1_1_Clement_of_Rome_Introduction	1	0.01587
6_Vol_1_1_Clement_of_Rome	3	0.00558
7_Vol_1_2_1_Mathetes_Introduction	0	0.00000
8_Vol_1_2_Mathetes	0	0.00000
9_Vol_1_3_a_1_Polycarp_Epistle_to_the_Philippians_Introduction	1	0.03030
10_Vol_1_3_a_Polycarp_Epistle_to_the_Philippians	0	0.00000
11_Vol_1_3_b_1_Polycarp_Martyrdom_Introduction	1	0.04762
12_Vol_1_3_b_Polycarp_Martyrdom	2	0.01149

13_Vol_1_4_1_Ignatius_Epistles_Introduction	1	0.00855
14_Vol_1_4_a_Ignatius_Epistle_to_the_Ephesians	0	0.00000
15_Vol_1_4_b_Ignatius_Epistle_to_the_Magnesians	0	0.00000
16_Vol_1_4_c_Ignatius_Epistle_to_the_Trallians	0	0.00000
17_Vol_1_4_d_Ignatius_Epistle_to_the_Philadelphians	0	0.00000
18_Vol_1_4_e_Ignatius_Epistle_to_the_Romans	0	0.00000
19_Vol_1_4_f_Ignatius_Epistle_to_the_Smyrnæans	0	0.00000
20_Vol_1_4_g_Ignatius_Epistle_to_the_Polycarp	2	0.01408
21_Vol_1_4_h_1_Ignatius_Syriac_Introduction	0	0.00000
22_Vol_1_4_h_Ignatius_Syriac	0	0.00000
23_Vol_1_4_i_1_Ignatius_Spurious_Epistles_Introduction	0	0.00000
24_Vol_1_4_i_Ignatius_Spurious_Epistles	0	0.00000
25_Vol_1_4_j_1_Ignatius_Martyrdom_Introduction	0	0.00000
26_Vol_1_4_j_Ignatius_Martyrdom	0	0.00000
27_Vol_1_5_1_Barnabas_Introduction	1	0.02326
28_Vol_1_5_Barnabas	0	0.00000
29_Vol_1_6_1_Papias_Introduction	1	0.03030
30ol_1_6_Papias	4	0.02235
31_Vol_1_7_1_Justin_Martyr_Introduction	1	0.01250
32_Vol_1_7_a_Justin_Martyr_First_Apology	4	0.00386
33_Vol_1_7_b_Justin_Martyr_Second_Apology	2	0.01408
34_Vol_1_7_c_Justin_Martyr_Dialouge_with_Trypho	0	0.00000
35_Vol_1_7_d_Justin_Martyr_Discourse_to_the_Greeks	1	0.01563
36_Vol_1_7_e_Justin_Martyr_Hortatory_Address_to_the_Greeks	4	0.00637
37_Vol_1_7_f_Justin_Martyr_On_the_Sole_Government_of_God	1	0.00943

38_Vol_1_7_g_Justin_Martyr_On_the_Resurrection	2	0.02000
39_Vol_1_7_h_Justin_Martyr_Other_Fragments	2	0.02326
40_Vol_1_7_i_Justin_Martyr_Martydom	0	0.00000
41_Vol_1_8_a_1_Irenaeus_Against_Heresies_Introduction	1	0.01064
42_Vol_1_8_a_Irenaeus_Against_Heresies	18	0.00169
43_Vol_1_8_b_Irenaeus_Fragments	0	0.00000
TOTAL	**53**	**0.00237**

Table 18: Third and final Ambiguous run with expanding co-occurrence window on test data

Finally, we reach the third and final run to catch any remaining tokens. This final run did in fact find a few remaining matches that else would have been left as false negatives. In testing, 53 candidate tokens -- or 0.24% -- where matched on this final run. It is up to the user if one wants to include this step. It is acknowledged that there is a considerable performance decrease in having to run a third run while only increasing the total tokens by less than one percent.

Text	Function	Total Tokens	Percentage	Running Percentage
All (Total)	calculateCertain()	13939	0.62233	62.2233%
	+surname	294	0.01313	63.546%
	+Run 1	3477	0.15524	79.070%
	+Run 2 (Expand)	4635	0.20694	99.764%
	+Run 3 (Expand)	53	0.00237	100%

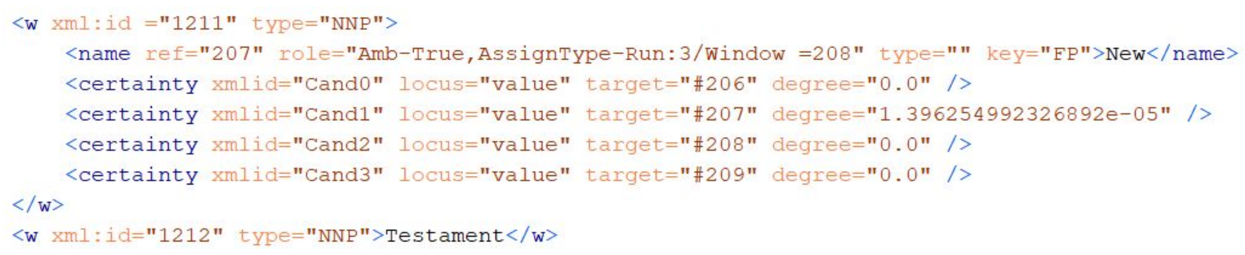
Table 19: Combined results of all algorithm components on test data

Table 19 demonstrates on a corpus level each step in the named entity disambiguation process. Of the four steps in the named entity disambiguation process, the second run consistently -- and on the average -- matched the most candidates. However, it is important to note that without the certain calculation, the surname calculation, and the first run of ambiguous calculations, this run would not have the appropriate connections to calculate the matches. As mentioned above, it is up to the end user as to the need for a third run. The cost-benefit of taking the time to run through the text a third time versus obtaining an additional 0.24 percent matches depends upon the specific end user requirements.

Issues with Examples

Issue 1: False Positive: Incorrect "Name" without "Surname

One issue responsible for many False Positives involves a "Name" in the knowledge base that must have a corresponding "Surname" in order for the match to be correct. For instance, take the entity "New York". Given the surname logic of the algorithm, the method will make a match if the word "New" and "York" are within three tokens of each other. However, what happens if the word "New" shows up without the surname? The algorithm treats this as a "name" and uses the corresponding logic to determine if it is Ambiguous or Unambiguous. In this case, "New" is seen as ambiguous. Here is an example where "New Testament" was found and "New" processed as a False Positive:



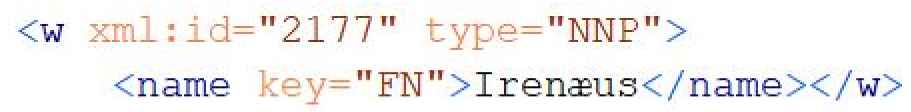
```
<w xml:id ="1211" type="NNP">
    <name ref="207" role="Amb-True,AssignType-Run:3/Window =208" type="" key="FP">New</name>
    <certainty xmlid="Cand0" locus="value" target="#206" degree="0.0" />
    <certainty xmlid="Cand1" locus="value" target="#207" degree="1.396254992326892e-05" />
    <certainty xmlid="Cand2" locus="value" target="#208" degree="0.0" />
    <certainty xmlid="Cand3" locus="value" target="#209" degree="0.0" />
</w>
<w xml:id="1212" type="NNP">Testament</w>
```

Figure 14: Example of a False Positive when using Surname logic

Here, token 1211 was matched as a possible candidate. The program believes that there are four possible matches. These four entities are in the knowledgebase from the training data. Specifically, the Gutenberg project has legal language that names all fifty states; these four are: New Hampshire (ID: 206), New Jersey (ID: 208), New Mexico (ID: 207), and New York (ID: 209). Therefore, the computer, using connection data in the tensor from previous runs, thinks that this "New" is actually entity 207, or New Mexico. In fact, this "New" is not part of a person or place, but the "New Testament". Note that this was matched during "Run: 3/Window 208". This means that the computer did not have a high confidence in tagging this as an entity. Run 3 is a last ditch effort so to speak and the window was increased from a default of 13 (average sentence length / 2). Therefore, future research should look into a threshold of certainty before assigning candidates.

Issue 2: False Negative- No connections within the window

Another issue involves False Negatives. The vast majority of False Negatives are caused by a lack of the known entity in the knowledge base. However, there are cases where a known entity in the knowledgebase is not tagged in the text. For instance, take the following figure:



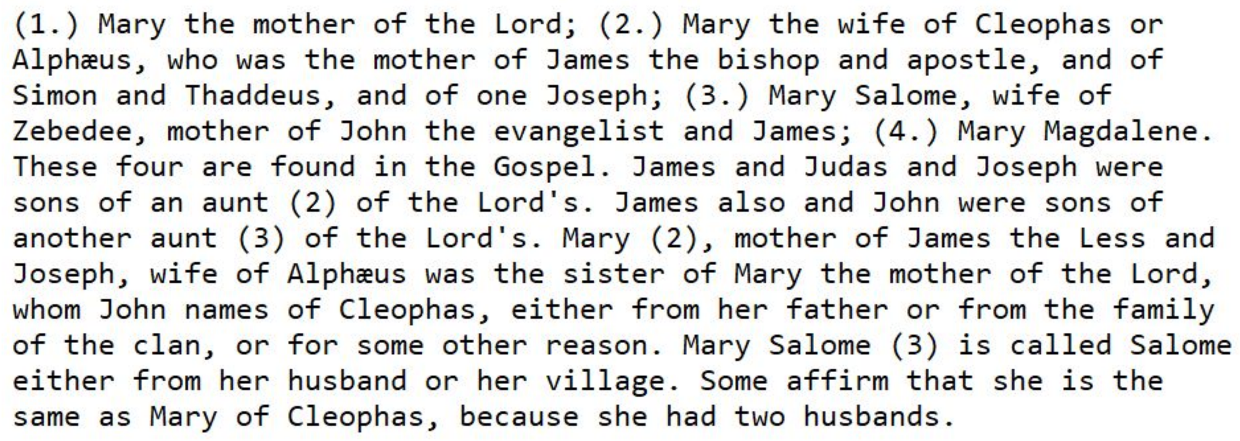
```
<w xml:id="2177" type="NNP">
    <name key="FN">Irenæus</name></w>
```

Figure 15: Example of False Negative

Irenæus was correctly assigned sixteen times before token 2177 and nine times after token 2177 in the same text. Therefore, why was token 2177 a False Negative? The answer lies in the co-occurrence window. In training, it was determined to set a limit on the window growth in runs 2 and 3. The reason: if you do not set a growth window, then eventually the process will run the score based on the entire text. This is not optimal in our domain of early Christian writings where multiple entities with the same name show up in the same text. Therefore, in this case, since there are two entities with the name "Irenæus," the algorithm looked for connections. In this case, there were no connections found within the original search window or the expanded search window in runs 2 and 3. Therefore, the entity went untagged and resulted in a False Negative. This could be solved in future research by either 1) expanding the co-occurrence window to the entire text, or; 2) defaulting to the entity with the highest tensor probability. Number one would default to the highest probability given the text entities while number two would default to the same entity each time.

Issue 3: Multiple Mentions of a Name in Same Context

One of the most difficult issues to solve is the usage of the same name within close proximity to each other. For example, take "Mary". There are multiple entities with the name "Mary" in the New Testament; there are even more when taking into account the Ante-Nicene writings. For instance, take this snippet from a footnote found in the Fragments of Papias:



```
(1.) Mary the mother of the Lord; (2.) Mary the wife of Cleophas or
Alphæus, who was the mother of James the bishop and apostle, and of
Simon and Thaddeus, and of one Joseph; (3.) Mary Salome, wife of
Zebedee, mother of John the evangelist and James; (4.) Mary Magdalene.
These four are found in the Gospel. James and Judas and Joseph were
sons of an aunt (2) of the Lord's. James also and John were sons of
another aunt (3) of the Lord's. Mary (2), mother of James the Less and
Joseph, wife of Alphæus was the sister of Mary the mother of the Lord,
whom John names of Cleophas, either from her father or from the family
of the clan, or for some other reason. Mary Salome (3) is called Salome
either from her husband or her village. Some affirm that she is the
same as Mary of Cleophas, because she had two husbands.
```

Figure 16: An example of multiple entities in same paragraph in the raw text

Here, one sees up to four different people named "Mary." Not only this, but there are also ambiguous entities used to describe each one: John, Joseph, and James. Using many of the current tools, they would have created a topic model of this text among all of the other texts and would have -- more than likely -- tagged all four of these as one "Mary". The current algorithm correctly identified two of the entities and incorrectly tagged two. One was a true False Positive while the other was due to "Mary Salome" not being included in the knowledgebase for analysis. This issue is a perfect case for future research into n-grams in use by other tools mentioned above.

Issue 4: False Negatives- Due to Missing Entity in the Knowledgebase

One of the biggest factors contributing to a lower F-score and Recall score happens when a token is tagged as a false negative due to a missing entity in the knowledgebase. If the "brain" is unaware of an entity, then it cannot tag the entity. While an index is a great resource to build a knowledge base, it is far from perfect. Indices are made by humans and humans can either: 1) make a mistake and miss an entity, or; 2) make a determination not to include an entity. While it is difficult to prove number one, a great argument can be made on point two in the case of the Ante-Nicene General Index. In volume ten, published after the first eight volumes of the texts, Dr. Bernhard Pick was tasked to create a General Index of the entire series. From the testing for this project, it appears that Dr. Pick made a decision to not index two areas: 1) the introduction material for each text, and; 2) Greek/Roman mythological entities found in both the writings of Justin Martyr and Irenaeus.

The evidence for this argument is found in analyzing the Recall scores of both the introductory material and the writings of both Justin Martyr and Irenaeus. All of these have very low Recall scores in comparison to the other writers. In fact, the lowest Recall scores are Justin Martyr's "Discourse to the Greeks," (Recall: 0.65979) which contain numerous Greek/Roman mythological entities, and the introduction to Irenaeus's "Against Heresies" (Recall: 0.62411). The reason for the low Recall scores are numerous False Negatives.

FN Entity	Mentions	In KB	Reason for FN
Achelous	1	F	Not in KB
Ægyptus	1	F	Not in KB
Agamemnon	2	F	Not in KB
Ajax	1	F	Not in KB
Antiope	1	F	Not in KB
Atreus	2	F	Not in KB
Briseis	1	F	Not in KB
Chryseis	1	F	Not in KB
Danaë	1	T	Human Typo
Danaus	1	F	Not in KB
Dods	1	F	Not in KB
Hector	1	F	Not in KB
Ithacan	1	F	Not in KB

Jupiter	1	T	No connections w/in Window
Lyda	1	F	Not in KB
Melanippe	1	F	Not in KB
OEdipus	1	F	Not in KB
Ouranos	1	F	Not in KB
Paris	1	T	No connections w/in Window
Pelides	1	F	Not in KB
Potter	1	F	Not in KB
Procne	1	F	Not in KB
Telamon	1	F	Not in KB
Thyestes	2	F	Not in KB
Triesperon	1	F	Not in KB
Troy	1	F	Not in KB
Ulysses	2	F	Not in KB
Venus	2	T	No connections w/in Window

Table 20: False Negatives found in Justin Martyr's *Discourse to the Greeks*

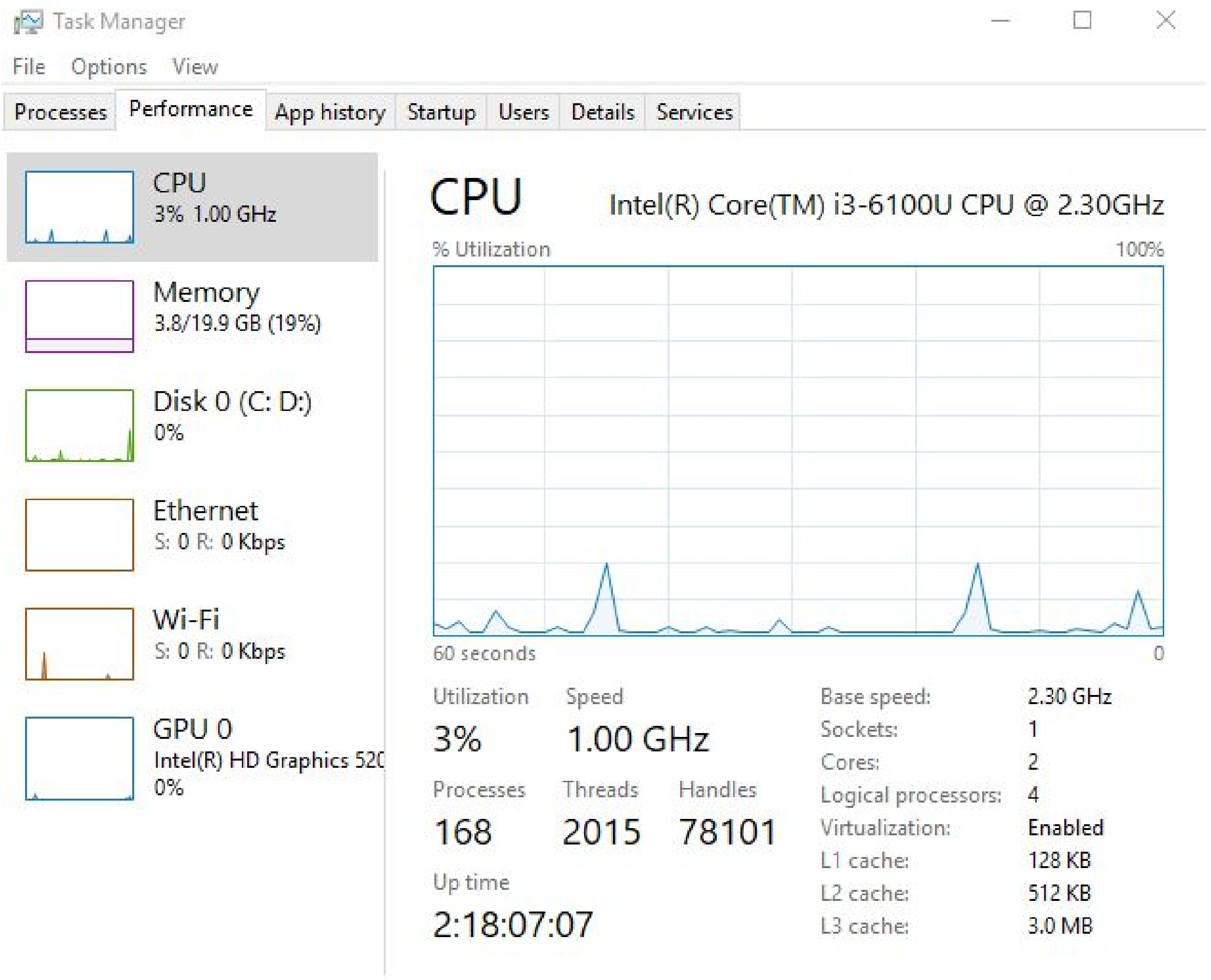
As seen from Table 20, 24 out of 28 False Negative entities were not found in the knowledgebase. One entity was misspelled in the knowledgebase due to human error during the creation of the comma delimited file, and three were False Negatives because they were ambiguous and could not make a connection within the window size.

If we assume that the computer would have tagged the 24 entities appropriately (True Positives) if they were included in the knowledgebase, then the Recall would have increased to 0.94949 with an F-score of 0.97409 (over the actual value of 0.79503). Therefore, the algorithm works quite nicely, but the method of using a General Index for the knowledge base comes at a cost: one is relying on the work of another and their subjective decisions on the quality of what makes it into the knowledgebase.

Performance Metrics

Using a tensor that grows with each text inherently comes with the foreknowledge of the fact that the more texts you run (therefore, the more data you insert into the tensor) the more of a performance hit one should see over time. On the same note, as with any system, the more data you insert into the tensor, the larger the system will get over time. Therefore, one must analyze the specific performance metrics of file space, memory consumption, and runtime to determine the feasibility of this process on their projects.

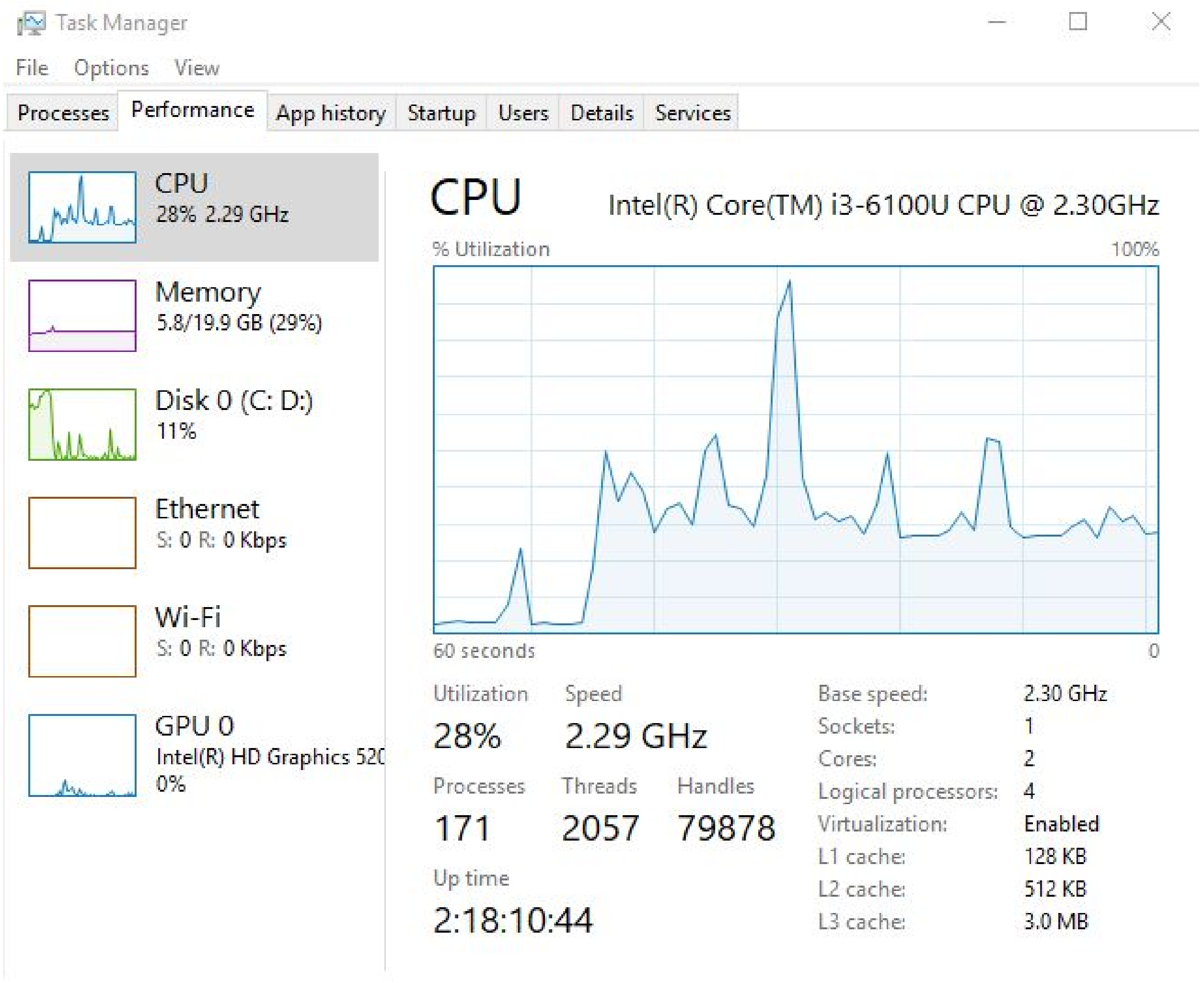
Disk Space and Memory Consumption



```
CPU
3% 1.00 GHz

Memory
3.8/19.9 GB (19%)

Disk 0 (C: D:)
0%

Ethernet
S: 0 R: 0 Kbps

Wi-Fi
S: 0 R: 0 Kbps

GPU 0
Intel(R) HD Graphics 520
0%

CPU    Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz

Utilization   Speed                Base speed:           2.30 GHz
3%            1.00 GHz             Sockets:              1
                                   Cores:                2
Processes   Threads   Handles      Logical processors:   4
168         2015      78101        Virtualization:       Enabled
                                   L1 cache:             128 KB
Up time                            L2 cache:             512 KB
2:18:07:07                         L3 cache:             3.0 MB
```

Figure 17: Computer Performance Before Running Program

Performance testing for this program occurred on a single core Intel i3-6100U 2.30GHz processor with 20 GB of RAM. Baseline measurements before kicking off the program are seen in Figure 17 above. As one will see, there are four different phases of the program that will impact performance: 1) creating/loading the tensor (only on first text); 2) processing the text file and knowledge base file; 3) ambiguous candidate processing, and; 4) saving the results to the tensor on disk. The following performance metrics are based on a 100 text tensor and a knowledge base of up to 3,150 entities.



CPU 28% 2.29 GHz
Memory 5.8/19.9 GB (29%)
Disk 0 (C: D:) 11%
Ethernet S: 0 R: 0 Kbps
Wi-Fi S: 0 R: 0 Kbps
GPU 0 Intel(R) HD Graphics 520 0%

CPU Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz

Utilization 28%
Speed 2.29 GHz
Processes 171
Threads 2057
Handles 79878
Up time 2:18:10:44

Base speed:	2.30 GHz
Sockets:	1
Cores:	2
Logical processors:	4
Virtualization:	Enabled
L1 cache:	128 KB
L2 cache:	512 KB
L3 cache:	3.0 MB

Figure 18: Computer Performance During Tensor Creation/ Data Loading

The creation and loading of the tensor with the knowledge base data is a memory and disk intensive process. The knowledge base is stored to disk in a simple comma delimited file. This file must be read from disk and loaded into the empty 3,150 x 3,150 x 100 tensor, which is at this point being stored in memory. In order to keep the memory usage to a minimum, the default size of the tensor was changed from a float 64 default in numpy, to a float 16 bit. This is a substantial memory and disk size savings. For instance: 3,150 x 3,150 x 100 using 64 bit float would be approximately 8 GB in size. By switching to 16 bit float, we see that it is ¼ of the size, or 2GB. As mentioned previously and in the future research section, most of this 2GB (or almost 1.5 billion cells) are empty -- technically, 0 filled cells -- so at this point it is not the most efficient storage. However, given the scope of this tool and process, this is more than sufficient.

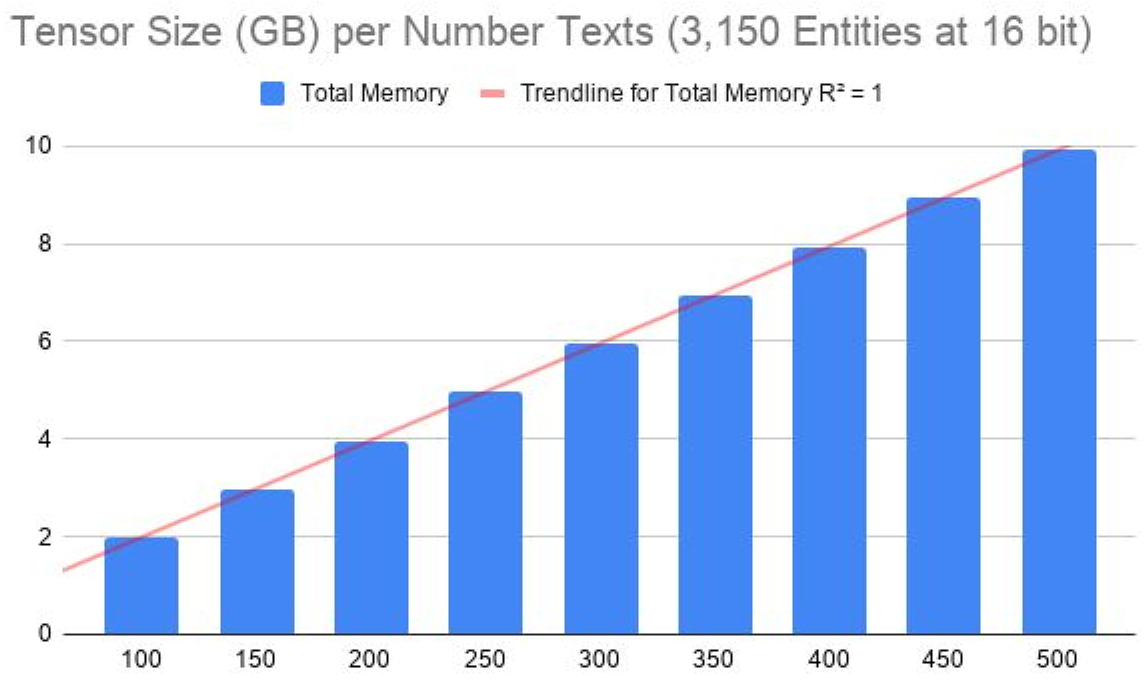


Figure 19: Tensor growth (in GB of RAM) as texts are added

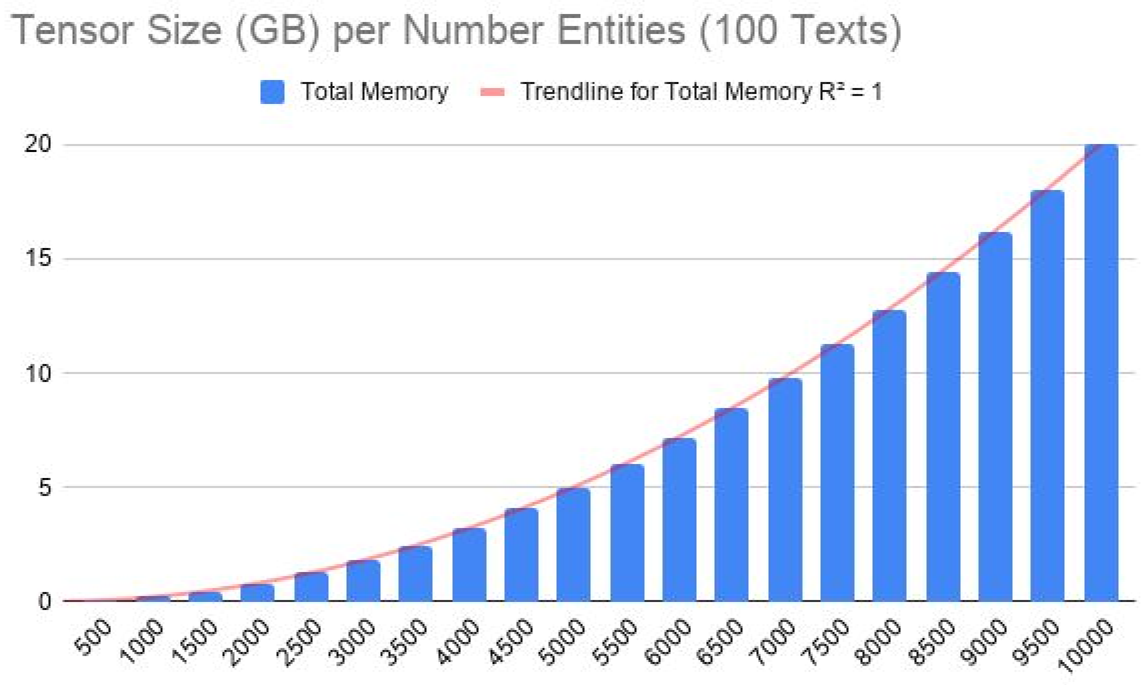
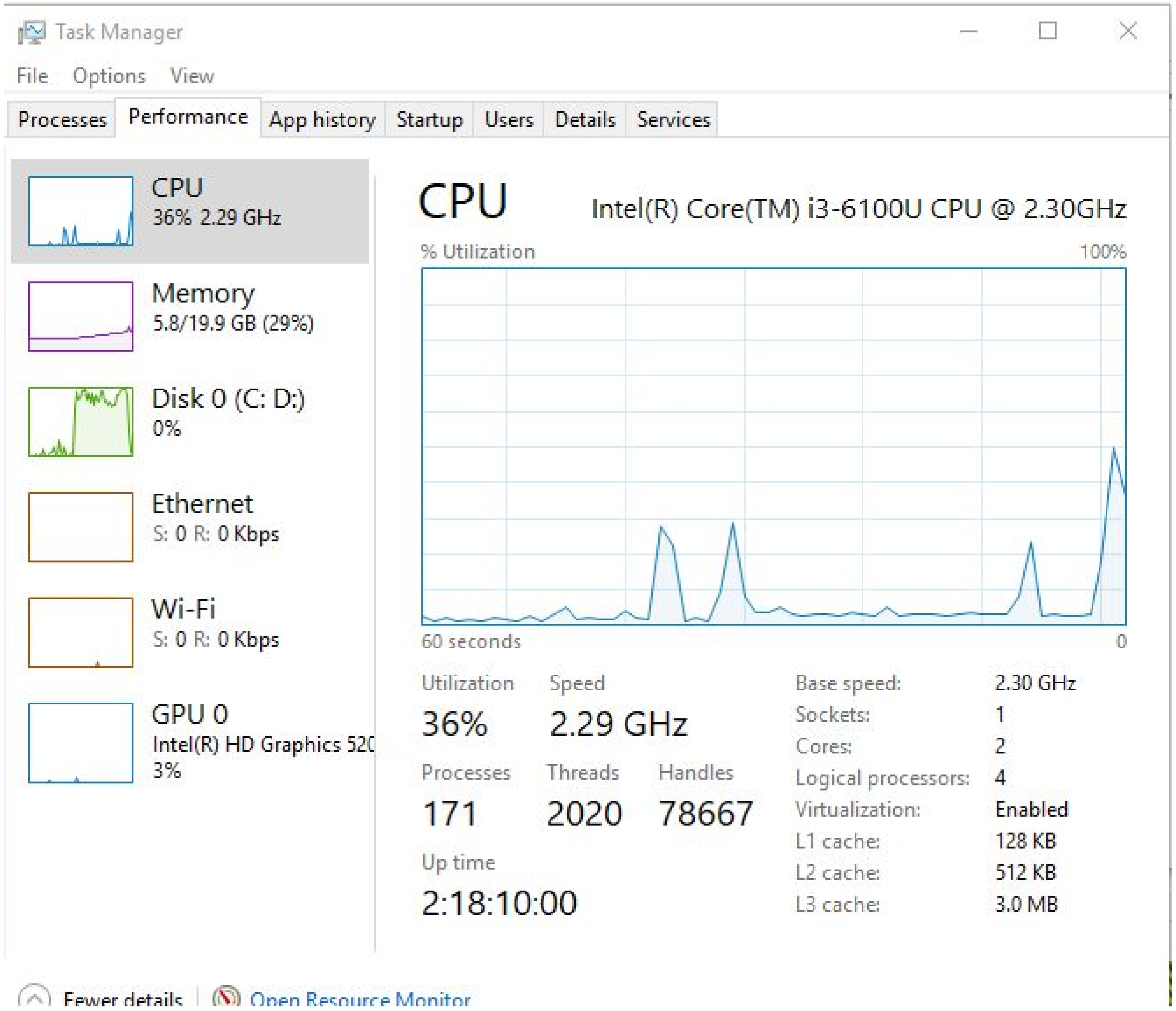


Figure 20: Tensor Size (in GB) per Number of Entities

While increasing the number of texts increases the tensor size in a linear fashion, increasing entity size in the knowledge base has a polynomial of second degree trend. This means that adding entities to the knowledge base will increase at a faster rate than adding texts. In our examples above, having a corpus of texts sized 100 with a knowledge base of 5,000 entities will produce a 5GB sized tensor. However, doubling the number of entities to 10,000 will create a tensor 20GB in size. Therefore, this tool is sufficient for domain specific projects, but probably not sufficient in its current state for big data (millions of entities).

As will be discussed in the future research section, there is ongoing research into this field. For instance, tensor decomposition tools, such as the SPLATT toolkit developed by the University of Minnesota, "...offers memory-efficient and operation-efficient algorithms to compute the CPD on large datasets, and supports hybrid parallelisms" [52]. CPD stands for and is “one of the prominent tensor decomposition formulations and commonly used to factorize a tensor as the summation of finite number of rank-one tensors” [52]. Therefore, the use of such tools may make this algorithm efficient for “big data” in the future.



CPU
36% 2.29 GHz

Memory
5.8/19.9 GB (29%)

Disk 0 (C: D:)
0%

Ethernet
S: 0 R: 0 Kbps

Wi-Fi
S: 0 R: 0 Kbps

GPU 0
Intel(R) HD Graphics 520
3%

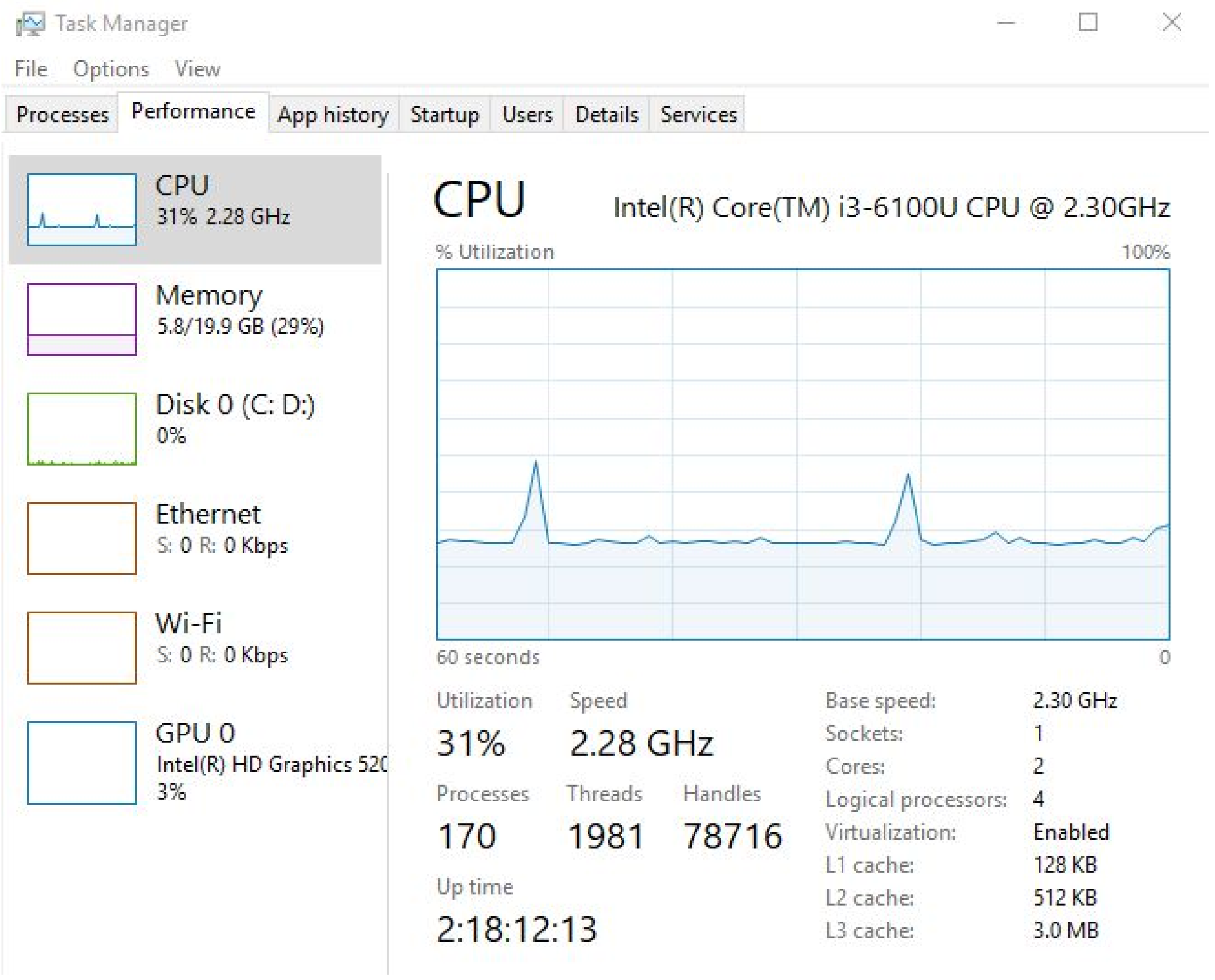
CPU Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz

Utilization 36%
Speed 2.29 GHz
Processes 171
Threads 2020
Handles 78667
Up time 2:18:10:00

Base speed:	2.30 GHz
Sockets:	1
Cores:	2
Logical processors:	4
Virtualization:	Enabled
L1 cache:	128 KB
L2 cache:	512 KB
L3 cache:	3.0 MB

Figure 21: Computer Performance End of Bulk Load

Once the tensor is created and loaded -- as seen above -- the user may run the program on multiple texts in batch, or as needed. If not running in batch, the user must load the tensor back into memory. As expected, this process is very disk intensive due to the fact that the computer must take the tensor file on disk and load it into memory. In our example above, the computer loads a 2 GB tensor thereby increasing the memory size from 3.8GB to 5.8GB and a spike in disk read/writes during the load. After the load is complete, the disk% goes down to 0%, but the memory usage stays at 5.8GB and is ready for data processing using the tensor data in RAM.

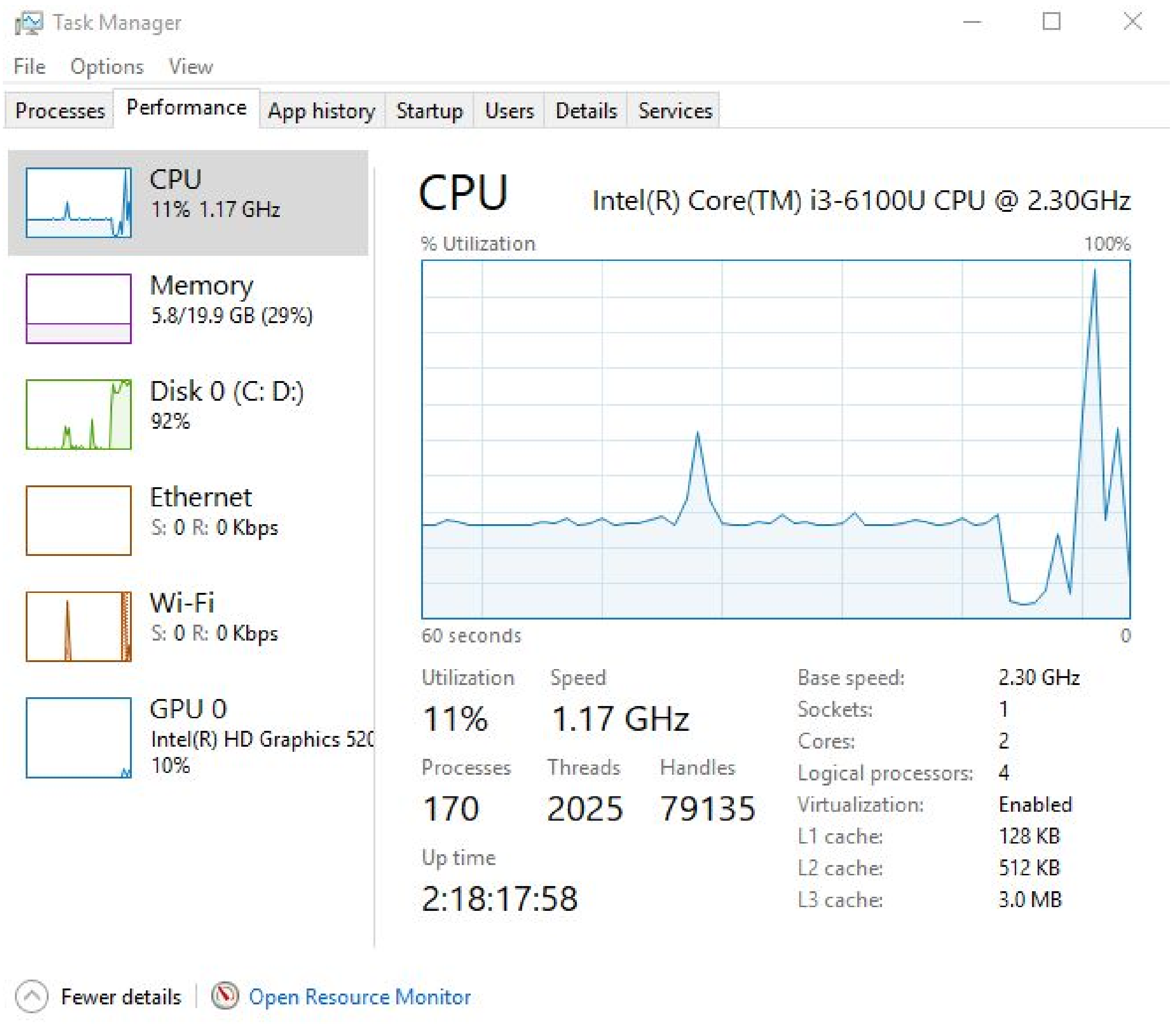


Utilization 31%
Speed 2.28 GHz
Processes 170
Threads 1981
Handles 78716
Up time 2:18:12:13

Base speed:	2.30 GHz
Sockets:	1
Cores:	2
Logical processors:	4
Virtualization:	Enabled
L1 cache:	128 KB
L2 cache:	512 KB
L3 cache:	3.0 MB

Figure 21: Performance During the Code Processing

Once the tensor is created and loaded, the program is ready for processing. As seen above in Figure 21, the processing of the ambiguous candidates is fairly straightforward. Memory is stable with the data loaded into RAM, there is no disk utilization since all calculations are performed in the memory, and the CPU percentage is slightly higher than the base before kicking off the program.



CPU
11% 1.17 GHz

Memory
5.8/19.9 GB (29%)

Disk 0 (C: D:)
92%

Ethernet
S: 0 R: 0 Kbps

Wi-Fi
S: 0 R: 0 Kbps

GPU 0
Intel(R) HD Graphics 520
10%

CPU Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz

% Utilization

Utilization 11%
Speed 1.17 GHz
Processes 170
Threads 2025
Handles 79135
Up time 2:18:17:58

Base speed:	2.30 GHz
Sockets:	1
Cores:	2
Logical processors:	4
Virtualization:	Enabled
L1 cache:	128 KB
L2 cache:	512 KB
L3 cache:	3.0 MB

Figure 23: Computer Performance Writing the Tensor to Disk from Memory

After the completion of the processing, we want to write the TEI file and save the tensor back to disk for persistent storage. Therefore, we see memory staying constant, but a spike in disk % during this process. At this point, the program is complete and all performance metrics go back to baseline.

Runtime

Understanding that the growth of the tensor expands with each text, it is a given that the runtime performance will degrade over time. However, will it degrade in such a fashion as to make the tool unusable in implementation on large corpora?

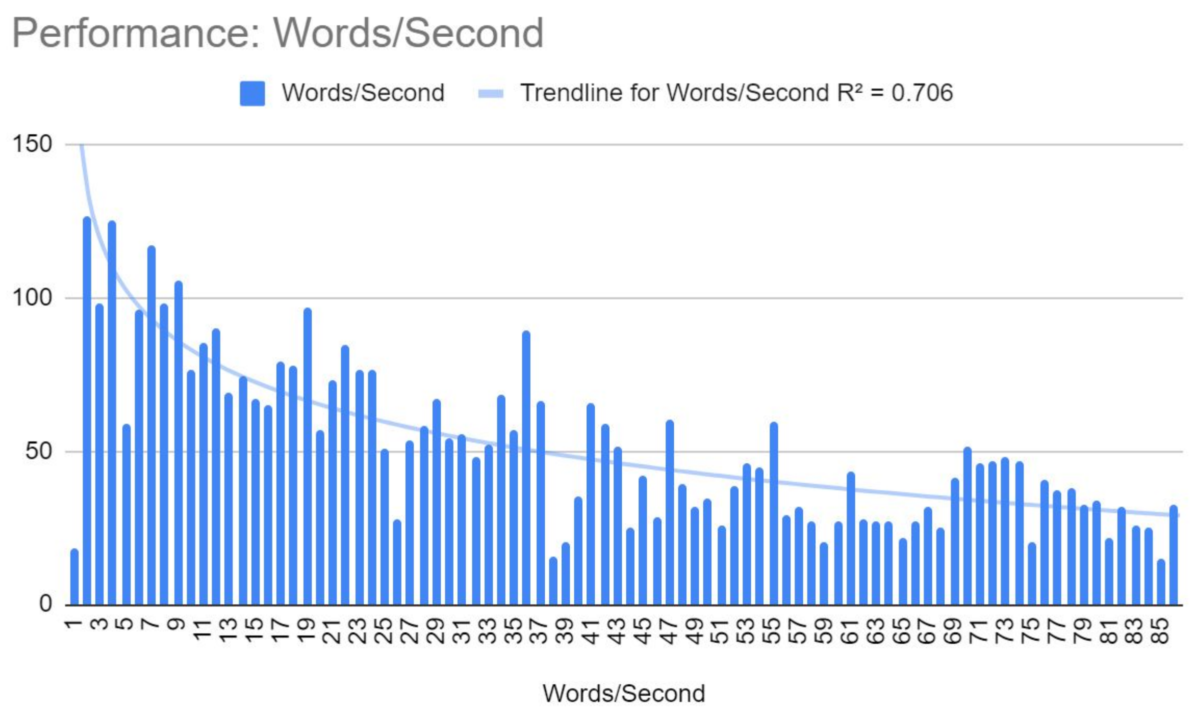


Figure 24: Performance measured in words/second

A measurement to test the runtime efficiency is how many words are in the text divided by the number of seconds needed for the program to complete the tagging process. As one sees in Figure 24, the performance does indeed degrade as the number of texts increases; this happens in a logarithmic fashion. This means that early in the process one will notice a rapid decrease in performance, but over time the curve will flatten out and the performance degradation will be minimal. During testing, the long-term degradation over 107 texts did not substantially cause any issues with the tagging of more than 5,000,000 words.

Other Issues

When dealing with datasets of this size, numpy would not work with 32 bit Python. Therefore, one must make sure to use 64 bit Python when performing large tensors in numpy.

Future Work

As mentioned throughout, there are numerous avenues for future research in this field and with this tool. For one, further implementing the feature that uploads unknown proper nouns into the tensor in an unsupervised fashion is ripe for consideration. Considerable effort will be needed to ensure that the NLP libraries find tokens that are true proper nouns. As noted previously, AllenNLP and BERT are both excellent candidates for solving this issue. By running a more sophisticated NLP process on the front end to tokenize and tag parts-of-speech, it is conceivable that an automated process may add unknown entities to the knowledgebase [57,56].

Another area that was tested, but did not provide any meaningful results is a function that takes into consideration the probability of each candidate's children with co-occurrence tokens' children. Say only one token is found within the 20 token window and it yields a low probability, can one then go and compare the probability of their children being found together in the text? This may yield some interesting results.

Incorporating more advanced knowledge base formats, such as RDF file, is an avenue to pursue. For instance, the syriaca.org project is currently creating RDF files full of relations and factoids about people, places, works, etc [53]. The ability for this algorithm to take advantage of this already produced supervised data would open up more collaboration with current scholarship. On this same line, incorporating the Pleiades dataset into the knowledgebase as well as the Syriaca.org dataset into the knowledgebase would add a considerable amount of entities from late antiquity [53, 54].

One avenue that would assist in incorporating these RDF files would be using the tensor to hold objects instead of just scoring values. Such advantages to this approach would allow one to house not only the connection between two entities, but what type of connection? Are they married? Are they brothers, sisters, spouses, friends, enemies, acquaintances, etc? It is feasible that creating a score based on type of relationship would add to the scoring mechanism of the algorithm.

On the same note, there are a lot of entities that current knowledge bases will not match in the text without human supervision. These are the use of pronouns in the text. Could one add to the algorithm an accurate way to assign pronouns to an entity in an unsupervised fashion? If so, there would be a great increase in the connections found in every text which, in theory, would increase the accuracy of the unsupervised approach.

Another area that is larger than just this algorithm is how to handle the size and scale needed when adding texts to a rank-3 tensor. As mentioned and demonstrated earlier, with each additional text, the tensor grows. While using 64 bit Python allows one to use larger tensors and changing the default tensor datatype to float 16 brings down the file size of the tensor, there is much more research to be done on the size. As an example, the vast majority of the tensor simply houses 0's due to no connections. This is one of the drawbacks to using the tensor approach for disambiguation, so research into efficient ways to use the tensor is needed. One method is to package the data in multiple tensors with X number of texts in each one and create a summed index of values. In theory, this will increase performance and decrease the size. However, this would have a negative impact if one needed to run and update any text (say, remove any false positives once someone looks at it through supervised means) or remove a text from the main tensor.

Conclusion

On the outset of this research, the goal was as follows: 1) to demonstrate that a simple knowledge base can be used to tag texts; 2) to demonstrate that a tensor can be used to efficiently and accurately tag entities in raw text documents, and; 3) create and provide a machine learning digital humanities tool for scholars interested in early Christian documents in English and Greek from late-antiquity. Given the results of the testing sets, all three goals are a success.

First, the research demonstrates that a simple knowledge base file in comma delimited format can be created quickly and easily by non-tech savvy persons. This knowledgebase, while simple, provides effective data for the algorithm to tag the texts. These knowledge bases can be created from corpora indices (such as the Ante-Nicene General Index), domain specific dictionaries (such as Eaton's Dictionary), or manually by reading through texts (as demonstrated in using the New Testament Gospel texts). One can expand the knowledge base to include as many alias fields as needed and as many entities/connections as needed. One can even connect texts to entities for a more accurate process, if the data is available.

Second, the research demonstrates that a tensor can indeed be used to efficiently and accurately tag entities in raw text documents. Our training and testing results indicate that an F-score in the mid-to-upper 90's is possible using the tensor based solution. Comparing to other research tools shows that this is an improvement over using existing global knowledge base setups with various methods for disambiguation. As mentioned earlier, this research does take note of the fact that the tensor based solution is a rapidly growing storage and memory solution. However, further research is demonstrating possibilities of making this more efficient for "big data".

Finally, the goal was to develop a tool for digital humanities scholars -- specifically those in early Christian resources of late antiquity -- to perform research in their field. Our testing indicates that there now exists a knowledge base file, a tagging tool, and an XML database suitable for early Christian historians and theologians to use -- and expand -- for research in the field.

REFERENCES

[1] “Study shows 8.7% decline in humanities bachelor’s degrees in 2 years.” [Online]. Available: https://www.insidehighered.com/news/2016/03/14/study-shows-87-decline-humanities-bachelors-degrees-2-years. [Accessed: 10-Aug-2019].

[2] Y. L. Zaila and D. Montesi, “Geographic Information Extraction, Disambiguation and Ranking Techniques,” in Proceedings of the 9th Workshop on Geographic Information Retrieval, New York, NY, USA, 2015, pp. 11:1–11:7.

[3] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, “DBpedia spotlight: shedding light on the web of documents,” presented at the Proceedings of the 7th International Conference on Semantic Systems, 2011, pp. 1–8.

[4] Z. Zheng, X. Si, F. Li, E. Y. Chang, and X. Zhu, “Entity Disambiguation with Freebase,” in Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01, Washington, DC, USA, 2012, pp. 82–89.

[5] B. Alex, K. Byrne, C. Grover, and R. Tobin, “Adapting the Edinburgh Geoparser for Historical Georeferencing,” Journal of Humanities & Arts Computing: A Journal of Digital Humanities, vol. 9, no. 1, pp. 15–35, Mar. 2015.

[6] F. Melo and B. Martins, “Automated Geocoding of Textual Documents: A Survey of Current Approaches,” Trans. in GIS, p. n/a–n/a, Jun. 2016.

[7] Y. L. Zaila and D. Montesi, “Geographic Information Extraction, Disambiguation and Ranking Techniques,” in Proceedings of the 9th Workshop on Geographic Information Retrieval, New York, NY, USA, 2015, pp. 11:1–11:7.

[8] A. Spitz, J. Geiß, and M. Gertz, “So Far Away and Yet So Close: Augmenting Toponym Disambiguation and Similarity with Text-based Networks,” in Proceedings of the Third International ACM SIGMOD Workshop on Managing and Mining Enriched Geo-Spatial Data, New York, NY, USA, 2016, pp. 2:1–2:6.

[9] M. B. Habib and M. van Keulen, “Improving Toponym Extraction and Disambiguation Using Feedback Loop,” in Web Engineering, M. Brambilla, T. Tokuda, and R. Tolksdorf, Eds. Springer Berlin Heidelberg, 2012, pp. 439–443.

[10] M. B. Habib and M. van Keulen, “A Hybrid Approach for Robust Multilingual Toponym Extraction and Disambiguation,” in Language Processing and Intelligent Information Systems, M. A. Kłopotek, J. Koronacki, M. Marciniak, A. Mykowiecka, and S. T. Wierzchoń, Eds. Springer Berlin Heidelberg, 2013, pp. 1–15.

[11] M. Badieh Habib Morgan and M. van Keulen, “Named Entity Extraction and Disambiguation: The Missing Link,” in Proceedings of the Sixth International Workshop on Exploiting Semantic Annotations in Information Retrieval, New York, NY, USA, 2013, pp. 37–40.

[12] L. Moncla, W. Renteria-Agualimpia, J. Nogueras-Iso, and M. Gaio, “Geocoding for Texts with Fine-grain Toponyms: An Experiment on a Geoparsed Hiking Descriptions Corpus,” in Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, New York, NY, USA, 2014, pp. 183–192.

[13] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M. (eds.). A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231.

[14] S. J. Wolf, A. Henrich, and D. Blank, “Characterization of Toponym Usages in Texts,” in Proceedings of the 8th Workshop on Geographic Information Retrieval, New York, NY, USA, 2014, pp. 7:1–7:8.

[15] X. Zhong, J. Liu, Y. Gao, and L. Wu, “Analysis of co-occurrence toponyms in web pages based on complex networks,” Physica A: Statistical Mechanics and its Applications, vol. 466, pp. 462–475, Jan. 2017.

[16] A. Eckhardt, J. Hreško, J. Procházka, and O. Smrſ, “Entity Linking Based on the Co-occurrence Graph and Entity Probability,” in Proceedings of the First International Workshop on Entity Recognition & Disambiguation, New York, NY, USA, 2014, pp. 37–44.

[17] R. Wiener, Y. Ben-Simhon, and A. Chen, “AOL’s Named Entity Resolver: Solving Disambiguation via Document Strongly Connected Components and Ad-Hoc Edges Construction,” in Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, 2016, pp. 595–596.

[18] Y. Zhang, F. Zhang, P. Yao, and J. Tang, “Name Disambiguation in AMiner: Clustering, Maintenance, and Human in the Loop.,” in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, New York, NY, USA, 2018, pp. 1002–1011.

[19] A. Alhelbawy and R. Gaizauskas, “Named Entity Disambiguation Using HMMs,” in Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 03, Washington, DC, USA, 2013, pp. 159–162.

[20] H. M. Wallach, D. M. Mimno, and A. McCallum, “Rethinking LDA: Why Priors Matter,” in Advances in Neural Information Processing Systems 22, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1973–1981.

[21] A. Pilz and G. Paaß, “From Names to Entities Using Thematic Context Distance,” in Proceedings of the 20th ACM International Conference on Information and Knowledge Management, New York, NY, USA, 2011, pp. 857–866.

[22] “Topic Modeling.” [Online]. Available: http://mallet.cs.umass.edu/topics.php. [Accessed: 10-Aug-2019].

[23] L. Duan et al., “An Online Name Disambiguation Method Based on Entity and Property Co-occurrence,” in 2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE), 2017, pp. 122–125.

[24] D. Moussallem, R. Usbeck, M. Röeder, and A.-C. N. Ngomo, "MAG: A Multilingual, Knowledge-base Agnostic and Deterministic Entity Linking Approach," in Proceedings of the Knowledge Capture Conference, New York, NY, USA, 2017, pp. 9:1–9:8.

[25] "Semantic Web Machine Reading with FRED | www.semantic-web-journal.net." [Online]. Available: http://www.semantic-web-journal.net/content/semantic-web-machine-reading-fred-1. [Accessed: 10-Aug-2019].

[26] "GERBIL — Agile Knowledge Engineering and Semantic Web (AKSW)." [Online]. Available: http://aksw.org/Projects/GERBIL.html. [Accessed: 20-Jun-2019].

[27] M. Cornolti, P. Ferragina, and M. Ciaramita, "A Framework for Benchmarking Entity-annotation Systems," in Proceedings of the 22Nd International Conference on World Wide Web, New York, NY, USA, 2013, pp. 249–260.

[28] G. Munnelly and S. Lawless, "Investigating Entity Linking in Early English Legal Documents," in Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries, New York, NY, USA, 2018, pp. 59–68.

[29] Á. L. Garrido, S. Ilarri, S. Sangiao, A. Gañán, A. Bean, and Ó. Cardiel, "NEREA: Named Entity Recognition and Disambiguation Exploiting Local Document Repositories," in 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), 2016, pp. 1035–1042.

[30] I. Lasek and P. Vojtás, "Various approaches to text representation for named entity disambiguation," International Journal of Web Information Systems; Bingley, vol. 9, no. 3, pp. 242–259, 2013.

[31] H. Zhang, W. Zhang, T. Huang, X. Liang, and K. Fu, "A Two-Stage Joint Model for Domain-Specific Entity Detection and Linking Leveraging an Unlabeled Corpus," Information; Basel, vol. 8, no. 2, p. 59, 2017.

[32] M. Arenas and J. Pérez, "Querying Semantic Web Data with SPARQL," in Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, New York, NY, USA, 2011, pp. 305–316.

[33] "TEI: Text Encoding Initiative." [Online]. Available: http://www.tei-c.org/index.xml. [Accessed: 23-Oct-2016].

[34] “Extensible Markup Language (XML).” [Online]. Available: https://www.w3.org/XML/. [Accessed: 23-Oct-2016].

[35] “eXistdb - The Open Source Native XML Database.” [Online]. Available: http://www.exist-db.org/exist/apps/homepage/index.html. [Accessed: 23-Oct-2016].

[36] RDF repository for all Syriaca.org data. This repository is based on the development branch of https://github.com/srophe/srophe-app-data and should be considered beta. : srophe/srophe-data-rdf. Syriaca.org: The Syriac Reference Portal, 2019.

[37] Anonymous, The Bible, King James version, Book 40: Matthew. Project Gutenberg, http://www.gutenberg.org/cache/epub/8040/pg8040.txt, 2005.

[38] Anonymous, The Bible, King James version, Book 41: Mark. Project Gutenberg, http://www.gutenberg.org/cache/epub/8041/pg8041.txt, 2005.

[39] Anonymous, The Bible, King James version, Book 42: Luke. Project Gutenberg, http://www.gutenberg.org/cache/epub/8042/pg8042.txt, 2005.

[40] Anonymous, The Bible, King James version, Book 43: John. Project Gutenberg, http://www.gutenberg.org/cache/epub/8043/pg8043.txt, 2005.

[41] “The New Testament according to the Vatican Manuscript,” Book of Matthew. Project Gutenberg, http://www.gutenberg.org/files/31802/31802-0.txt, 2010.

[42] M. G. Easton, Illustrated Bible Dictionary, and Treasury of Biblical History, Biography, Geography, Doctrine, and Literature. T. Nelson, 1894.

[43] B. M. Metzger and M. D. Coogan, Eds., The Oxford Guide to People & Places of the Bible. Oxford: Oxford University Press, 2004.

[44] M. Anthony, The Sayings of the Desert Fathers: The Alphabetical Collection, Revised edition. Kalamazoo, Mich: Liturgical Press, 1984.

- Note: Used with permission of publisher

[45] “Christian Classics Ethereal Library.” [Online]. Available: http://www.ccel.org/. [Accessed: 27-May-2018].

[46] “Welcome to Python.org,” Python.org. [Online]. Available: https://www.python.org/. [Accessed: 23-Oct-2016].

[47] “NumPy — NumPy.” [Online]. Available: https://www.numpy.org/. [Accessed: 11-Aug-2019].

[48] “Hickle Documentation.” [Online]. Available: http://telegraphic.github.io/hickle/toc.html. [Accessed: 14-Aug-2019].

[49] “Home page for the PyPDF2 project.” [Online]. Available: http://mstamy2.github.io/PyPDF2/. [Accessed: 10-Aug-2019].

[50] “About — Classical Language Toolkit documentation.” [Online]. Available: http://docs.cltk.org/en/latest/about.html. [Accessed: 10-Aug-2019].

[51] C. M. Bishop, Pattern Recognition and Machine Learning. New York: Springer, 2011.

[52] Y. Tan and T. Imamura, “Performance Evaluation of a Toolkit for Sparse Tensor Decomposition,” in Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing, New York, NY, USA, 2018, pp. 5–6.

[53] D. Michelson, editor. (2016, October 22). Syriaca.org: Syriac Reference Portal [Online]. Available: http://syriaca.org/about-syriac.html

[54] R. Bagnall et al., “Pleiades: A community-built gazetteer and graph of ancient places,” 2006. [Online]. Available: http://pleiades.stoa.org. [Accessed: 23-Oct-2016].

[55] “Natural Language Toolkit — NLTK 3.4.4 documentation.” [Online]. Available: https://www.nltk.org/. [Accessed: 10-Aug-2019].

[56] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” arXiv:1810.04805 [cs], Oct. 2018.

[57] M. E. Peters et al., “Deep contextualized word representations,” arXiv:1802.05365 [cs], Feb. 2018.