GENEALOGY EXTRACTION AND TREE GENERATION FROM FREE FORM TEXT

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Timothy Sui-Tim Chu

December 2017

COMMITTEE MEMBERSHIP

TITLE:	Genealogy Extraction and Tree Generation from Free Form Text
AUTHOR:	Timothy Sui-Tim Chu
DATE SUBMITTED:	December 2017
COMMITTEE CHAIR:	Foaad Khosmood, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Franz Kurfess, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Lubomir Stanchev, Ph.D. Professor of Computer Science

ABSTRACT

Genealogy Extraction and Tree Generation from Free Form Text

Timothy Sui-Tim Chu

Genealogical records play a crucial role in helping people to discover their lineage and to understand where they come from. They provide a way for people to celebrate their heritage and to possibly reconnect with family they had never considered. However, genealogical records are hard to come by for ordinary people since their information is not always well established in known databases. There often is free form text that describes a person's life, but this must be manually read in order to extract the relevant genealogical information. In addition, multiple texts may have to be read in order to create an extensive tree.

This thesis proposes a novel three part system which can automatically interpret free form text to extract relationships and produce a family tree compliant with GEDCOM formatting. The first subsystem builds an extendable database of genealogical records that are systematically extracted from free form text. This corpus provides the tagged data for the second subsystem, which trains a Naïve Bayes classifier to predict relationships from free form text by examining the types of relationships for pairs of entities and their associated feature vectors. The last subsystem accumulates extracted relationships into family trees. When a multiclass Naïve Bayes classifier is used, the proposed system achieves an accuracy of 54%. When binary Naïve Bayes classifiers are used, the proposed system achieves accuracies of 69% for the child to parent relationship classifier, 75% for the spousal relationship classifier, and 73% for the sibling relationship classifier.

ACKNOWLEDGMENTS

This thesis would not have been possible without the help and support of those who have followed along with me on this incredible journey. Thank you so much:

- My parents, Kam and Mei.
- My siblings, Tiffany, Emily, and Terry.
- My girlfriend, Helen.
- My advisors, Dr. Khosmood, Dr. Kurfess, and Dr. Stanchev.
- All my friends and colleagues.
- The entire Computer Science Department at Cal Poly.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Genealogy

Genealogy, the study of families and the tracing of their lineages and history, dates back to historical times and was used extensively for tracking whether a person was of royal or common status [33]. Nowadays it is also conducted as a curious exercise in addition to learning one's birth rights. That curiosity, among other factors, has exploded genealogy into a two billion dollar industry, set to hit three billion by 2018 [10]. To conduct their research, genealogists use actual events, recorded events, oral interviews, historical records, genetic analysis, and other sources of data as the basic building blocks for making conclusions about the relationships among groups of people. They typically accumulate their findings in the form of family trees [2, 5]. A sample tree is shown in Figure 1.1. In these trees, the father and the mother are usually connected with a straight horizontal line and the children are placed below with a connecting arrow. This format allows visual separation of generations while still grouping together different sides of the family.

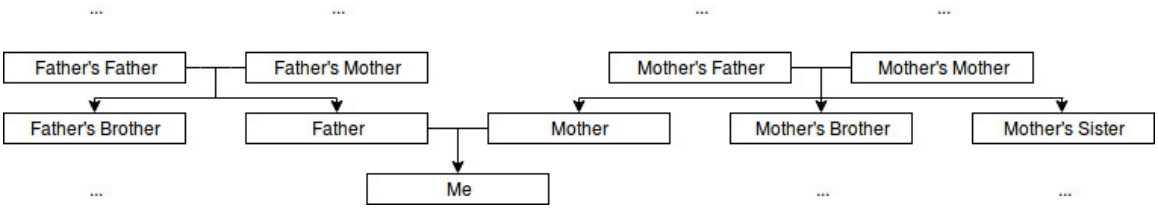


Figure 1.1: Family Tree Example

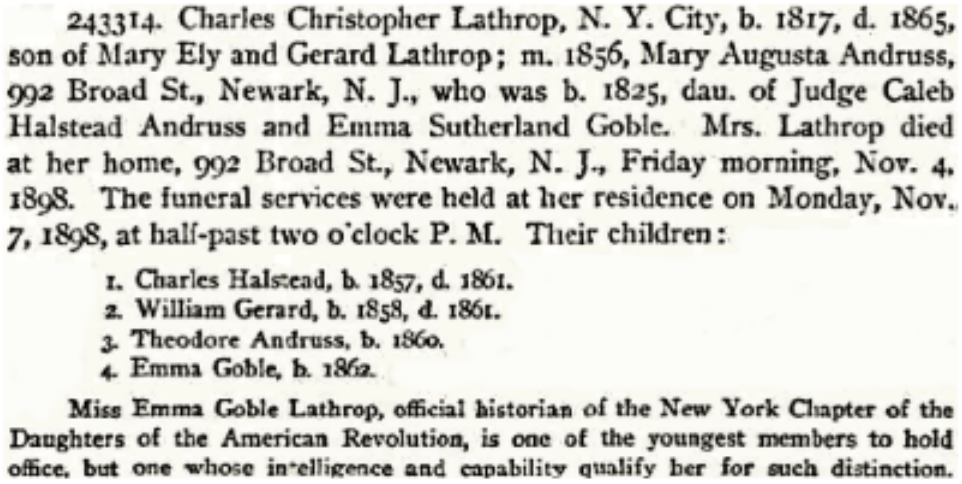
Presently, genealogy is still a heavily researched field because people are fueled by the desire and/or responsibility to learn their place in the larger historical picture, to preserve the present for the future generations, and to accurately tell the story of a

lineage [14]. In terms of the benefits for the user of genealogical records, people can discover their lineage and celebrate their heritage as well as reconnect with family members who they never knew or even considered. As such, this thesis aims to improve the field by increasing the availability of genealogical records for public use by tapping into the largely unprocessed domain of free form text.

1.2 Free Form Text

To preface the discussion on free form text and its relationship to this thesis, it is valuable to know that most genealogical records are gathered from well-defined data sources (see Figure 1.2). These include historical artifacts, such as censuses, social security death indices, church records, hospital records, birth certificates, marriage listings, etc [2]. These types of data sources all have their content in a consistent and organized manner and typically are recorded with the sole intent of documenting information. Take for example the birth certificate: there is an explicit underlined section for the father, the mother, and the new born child. It is not to say that these data sources are insufficient. On the contrary, they have provided the basis on which genealogy has grown thus far. However, there is a wealth of information available that does not fall in the same category. One such source is free form text.

For the remainder of this paper, free form text is defined as text with no formal structure and no pattern. A good example of this would be something like a story (See Figure 1.3) or a memoir. There is no explicit requirement to contain any genealogical relationships, and if there are any, they could be said in a wide variety of ways. The challenge of extracting genealogical records from free form text is that there is no easy way to match expressions, unlike the well-defined data sources. Revisiting the birth certificate example, an algorithm could be developed that pulls the father's, mother's, and child's name from the document. The procedure would always know



243314. Charles Christopher Lathrop, N. Y. City, b. 1817, d. 1865, son of Mary Ely and Gerard Lathrop; m. 1856, Mary Augusta Andruss, 992 Broad St., Newark, N. J., who was b. 1825, dau. of Judge Caleb Halstead Andruss and Emma Sutherland Goble. Mrs. Lathrop died at her home, 992 Broad St., Newark, N. J., Friday morning, Nov. 4, 1898. The funeral services were held at her residence on Monday, Nov. 7, 1898, at half-past two o'clock P. M. Their children:

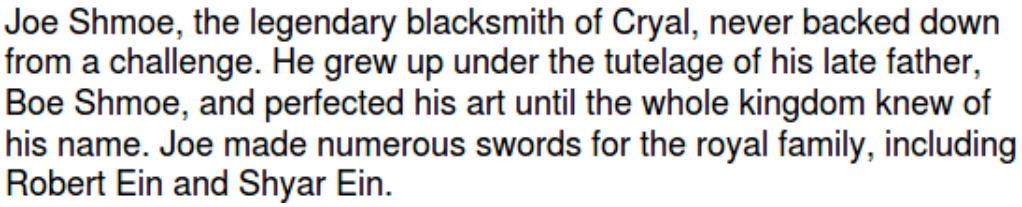
1. Charles Halstead, b. 1857, d. 1861.
2. William Gerard, b. 1858, d. 1861.
3. Theodore Andruss, b. 1860.
4. Emma Goble, b. 1862.

Miss Emma Goble Lathrop, official historian of the New York Chapter of the Daughters of the American Revolution, is one of the youngest members to hold office, but one whose intelligence and capability qualify her for such distinction.

Figure 1.2: Well-Defined Data Example

[23]

which underlined fields referred to whom, simply because the format of the birth certificate is consistent across all birth certificates. However, the task is considerably harder for free form text because the algorithm has to first understand the context and meaning of each word.



Joe Shmoe, the legendary blacksmith of Cryal, never backed down from a challenge. He grew up under the tutelage of his late father, Boe Shmoe, and perfected his art until the whole kingdom knew of his name. Joe made numerous swords for the royal family, including Robert Ein and Shyar Ein.

Figure 1.3: Free Form Text Example

As mentioned, the use of free form text in genealogy is a largely untouched domain. In addition, free form text is another source of data that is available for the ordinary person (who might not be included in a genealogical database compared to more renowned individuals, like a president, a celebrity, a sports star, etc). This thesis utilizes the English version of Wikipedia as its training source of free form text because it is the most readily available and verifiable source of free form text. Each Wikipedia

page has a link to a Wikidata page, which has human verified fields. A Wikipedia page about a person will have a Wikidata page which states who their parents are, who their spouse is, who their siblings are, and who their children are. It is important to note that there is not always a one-to-one match between the relationships on the Wikipedia page and the Wikidata page. As a result, some Wikipedia pages may specify a relationship in the article which is not captured in the Wikidata page, and some Wikidata pages may specify relationships in some of its fields which are not represented in the Wikipedia page. This will require some filtering of the data to ensure consistency, and is the nature of a tool like Wikipedia since it is crowd sourced and requires human effort to create and verify both the Wikipedia and Wikidata pages.

1.3 Assumptions

This thesis assumes that the English version of Wikipedia is an acceptable source of free form text to train on because there is no particular way in which the author has to state the facts. There are sections in Wikipedia pages that do provide some formatting, but again, it is up to the discretion of the human contributor to include what they want. To some effect, this provides us a loose guarantee that some articles will have genealogical information, given that the contributor does try to follow along with the sections (since some are naturally inclined to include genealogical information). Due to this variability, this thesis believes that the model will be extendable, to some degree, towards other free form texts as well. In addition, this thesis assumes that names follow the English convention of first name, last name. This assumption is needed when trying to locate the entities for the genealogical relationships and when performing anaphora resolution.

1.4 Contributions

The core problem this thesis tries to solve is whether there is a genealogical relationship between two entities found in English free form text, and if there is, whether that relationship is a child to parent relationship, a spousal relationship, or a sibling relationship. Specifically, this thesis has three main contributions to the field of genealogy:

1. A generic and extendable golden data set which can be utilized by other genealogists in their research on free form text.

2. A pipeline for converting free form text into feature vectors and training a Naïve Bayes machine learning model, all the way from text cleaning to named entity recognition and anaphora resolution.

3. A system for converting chosen relationships into a forest of family trees in the GEDCOM format.

1.5 Experiments

Multiple experiments are run to evaluate the accuracy of the machine learning model in predicting the relationships between two entities found in Wikipedia free form text. Table 1.1 contains a high level overview. Both multiclass and binary Naïve Bayes classifiers are tested against, with two variable parameters in the pipeline. The first is whether to look up or classify the gender during the anaphora resolution step and the second is whether to test on a balanced or unbalanced test set. Rationale for utilizing the Naïve Bayes classifier over other models will be discussed in Chapter 2, while pipelines and their intricacies will be discussed in Chapters 4 and 5. Results specifically for the machine learning model will be in Chapter 6.

Table 1.1: Experimental Information

Parameter	**Value**	**Notes**
Data Source	Wikipedia/Wikidata	Assumption of Free Form Text
Relationships Examined	Child to Parent Relationships, Spousal Relationships, Sibling Relationships, None	
Preprocessing Steps	Named Entity Recognition, Anaphora Resolution (with Gender Classification)	
Machine Learning Algorithm	Naïve Bayes Classifiers	Multiclass and Binary Classifiers
Options	Look up or Classify Gender and Balanced or Unbalanced Test Set	

1.6 Thesis Layout

The following chapter expands on the background of genealogy, GEDCOM, and the tools utilized in this thesis. Then, the thesis explores related work in the field. In subsequent chapters, the three part system and the experimental set up are described. Finally, the results of the model and concluding remarks wrap up the thesis.

Chapter 2

BACKGROUND

2.1 State of Genealogy

Currently, genealogy is focused on the domain of well-defined data. Most of the academic research and commercial efforts are focused on indexing digitized, well-defined records to make searchable databases. For example, in a statement released by FamilySearch.com in 2014, around 5 billion historical records have been digitized, with around 10 billion more to be processed. Their goal is to index those 5 billion records in the next twenty to thirty years [5, 6]. The process of indexing is done either manually or through some program. In the manual process, human volunteers typically look at a record that has been scanned online. They manually pull out relevant pieces of information and add it to the database that they are working on. The programs operate with the same goal, except the software is required to determine which pieces of information are relevant in the first place.

Some genealogy is also conducted in the realm of DNA testing. This has been done more recently, and is only possible through the advancement of biotechnology. There are three main kinds of DNA testing: Y-DNA testing, Mitochondrial DNA testing, and Autosomal DNA testing. The Y-DNA test is specific to males and looks specifically at the Y chromosome. The Mitochondrial DNA test is specific to females and reveals information from the mitochondrial DNA passed on by females. Lastly, the Autosomal DNA test examines autosomes (chromosomes other than the X and Y) to reveal ancestry [24]. The process of mapping a person's DNA has reached a state where it is cheap and scalable. The big competitors in this area are AncestryDNA, 23andMe, and FamilyTreeDNA [2, 1, 7].

2.2 Genealogical Databases

There are a large number of genealogical databases out there, prepared by companies, amateur genealogists, and researchers. The big competitors are found in Table 2.1.

Table 2.1: List of Genealogical Databases

1	Ancestry.com
2	FamilySearch.com
3	Genealogy.com
4	AmericanAncestors.org
5	UsGenWeb.com

2.2.1 Ancestry.com

Ancestry.com is the largest for-profit genealogy company that launched in 1996. It is based in Utah, United States. In a press release in July 2014, they noted that they provide access to almost 16 billion records, some of which were contributed by users of the site [2]. The majority of its records are from the United States, although more recent years have included records for other countries like Canada, the UK, and other European countries.

2.2.2 FamilySearch.com

FamilySearch.com is a genealogy organization that is run by The Church of Jesus Christ of Latter-day Saints, also known as the Mormon Church. It is also based in Utah, and actually provides free access to its resources and services. The main motive behind the service is to provide the necessary information to the members of

its church so that they can perform their religious duties. The majority of its records are in relation to the members of the church [5].

2.2.3 Genealogy.com

Genealogy.com is a for-profit company that started out with the well known software, Family Tree Maker. The site is now a search space for users to browse through digitized records. It has information from census, military, marriage, probate, church, and other records. They also have information from immigration records through sources like passenger lists, published works, etc [9].

2.2.4 AmericanAncestors.org

AmericanAncestors.org is a site run by the New England Historic Genealogical Society, the oldest and largest genealogical society in the United States. It mainly contains information about immigrants to New England. The site has records that pertain to the ancestors of the Mayflower [11].

2.2.5 UsGenWeb.com

UsGenWeb.com is a free to use site that is managed by a group of volunteers. It was established in 1996 with the purpose of creating an online center for genealogical research. They manage information for every state and county in the United States, collaborating with the individual states and counties. One of their big efforts is the Tombstone Project, which houses records containing gravestone transcriptions from cemeteries in the United States [12].

2.3 GEDCOM

GEDCOM, the acronym for Genealogical Data Communications, is a specification for capturing genealogical relationships from a family tree and is the chosen specification for the results of this thesis. It was developed by The Church of Jesus Christ of Latter-day Saints in 1984 and it is still the widely accepted format for saving and sending genealogical data [30]. The current standard is using version 5.5, but other branched varieties exists that are better fitted in different circumstances. Figure 2.1 shows an example GEDCOM for a family with a father, mother, and a child.

```
0 HEAD
1 CHAR ASCII
1 SOUR Example GEDCOM File
1 GEDC
2 VERS 5.5
2 FORM Lineage-Linked
1 SUBM @SUBMITTER@
0 @SUBMITTER@ SUBM
1 NAME Thesis

0 @FATHER@ INDI
1 NAME /Father/
1 SEX M
1 FAMS @FAMILY@

0 @MOTHER@ INDI
1 NAME /Mother/
1 SEX F
1 FAMS @FAMILY@

0 @CHILD@ INDI
1 NAME /Child/
1 FAMC @FAMILY@

0 @FAMILY@ FAM
1 HUSB @FATHER@
1 WIFE @MOTHER@
1 CHIL @CHILD@
0 TRLR
```

Figure 2.1: GEDCOM Example

Each GEDCOM file starts with some boilerplate information about things like a GEDCOM file id, a submitter, and a version number. This is followed by the actual genealogical information. Each person is marked by the expression: '0 @<person-id>@ INDI' and is followed by attributes like '1 NAME <name>', '1 SEX <M | F>', and '1 FAMS @<relationship-id>@'. Groups of relationships are marked by

the expression: '0 @<relationship-id>@ FAM' and are followed by attributes like '1 HUSB @<person-id>@', '1 WIFE @<person-id>@', and '1 CHIL @<person-id>@'.

While GEDCOM is the standard, there are some limitations. For example, poor support and integration for different sources, no support for formal conclusions using formal logic, and lack of structures other than lists [35]. Other alternatives to GEDCOM also exist, like the graph database model suggested by Perea [27]. However, many alternatives admit the dominance of GEDCOM and provide converters.

2.4 Tools

This thesis utilizes a variety of tools that will be contextualized in this section. The tools fall under four broad domains: Data Mining, Data Storage, Natural Language Processing, and Machine Learning.

2.4.1 Data Mining

The data mining portion of this thesis utilizes the Python *wikipedia* module for the scraping of free form text for training data from Wikipedia. This module, created by Wikipedia developers, has functionality for searching through Wikipedia, obtaining article summaries, and obtaining whole articles. Since the data of interest was actually the entire page, the functionality for obtaining the whole article was used. The API returns a *WikipediaPage* object, which has functions for accessing the entire article contents, HTML, images, links, references, summary, and numerous other pieces of information.

An important consideration for the data mining portion of this thesis was whether to utilize DBPedia, an association that already has available data sets of information mined from Wikipedia [4]. They do have a data set specifically for people that contains

genealogical relationships like father, mother, spouse, and children. However, the main factor discouraging its use was that there was no easy method to link the actual text the data was mined from. Additionally, that data set contained an excess of data regarding the person that would never be used and waste space. The data set created by this thesis is more lightweight and puts the spotlight on the text.

2.4.2 Data Storage

The data storage portion of this thesis utilizes MongoDB, a non-relational database system founded in 2007 that allows storage of JSON objects. Each database contains a number of collections, which is the equivalent of a table in SQL. Each record, called a document, has a primary key (that is assigned by MongoDB if no value is provided by the user), which allows for quick access through indexing. One problem faced with MongoDB is the lack of document locking. A common workaround was utilized, which essentially used a primary key write-lock strategy to lock documents so that concurrency could be achieved. Note that MongoDB was chosen over a traditional relational database system because of its ease of integration with Python dictionary objects, which can essentially be thought of as JSON objects. Additionally, JSON allows for lists to be stored.

2.4.3 Natural Language Processing

The natural language processing portion of this thesis makes use of two core concepts: Named Entity Recognition and Anaphora Resolution.

Named entity recognition is a process that locates important entities mentioned in text, like people, organizations, dates, holidays, locations, etc. It is useful in this thesis because a relationship requires separate individuals and named entity recognition can find those individuals. Consider Figure 2.2 below. The goal of named entity

recognition is to locate the four entities in the text: Joe Smith (a person), Jane Doe (a person), Memorial Day (a date) and Intuit (a company). The Stanford Natural Language Processing Group's *CoreNLP* library was chosen for this thesis because it is a top notch system that also provides a generic base upon which more rules can be added on. On the CoNLL 2003 data set, this system achieved a 93.28% precision, a 92.71% recall, and 92.99% F1-score. The library is capable of identifying locations, persons, organizations, money, percents, dates, and time. An alternative that was considered is *spaCy*, which is another open source natural language processing library. *SpaCy* is actually recognized as the fastest, competitive syntactic parser in the world, but extra functionality is harder to integrate into this library. *SpaCy* is capable of locating persons, organizations, facilities, locations, products, events, languages, and even works of art.

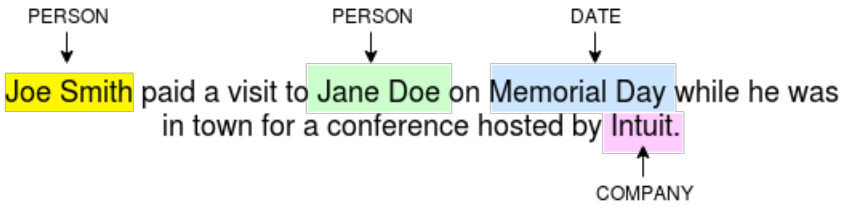


Figure 2.2: Named Entity Recognition Example

Anaphora resolution is a process that resolves reference to an earlier or later mentioned entity (which is usually discovered through named entity recognition) [26]. Consider Figure 2.3 below. The goal of anaphora resolution is to resolve the "he" and "Joe" to Joe Smith and the "they" to both Joe Smith and Jane Doe. This concept is a necessary step for preprocessing free form text before feature vectorization. Many different libraries were considered, the top contender being *Neural coref*, an open source anaphora resolution tool. It is a state-of-the-art coreference resolution (a subset of anaphora resolution) system that implements the system discussed in [15]. This system is based on deep learning with neural networks and showed significant

improved performance over other current techniques when first introduced in 2016. However, initial testing of *Neural coref* on Wikipedia training free form text showed that this system was not good enough. Thus, this thesis opted for a naive way of doing anaphora resolution, since most pronouns in the training free form text are referring to the article person. This is further discussed in Chapter 5.

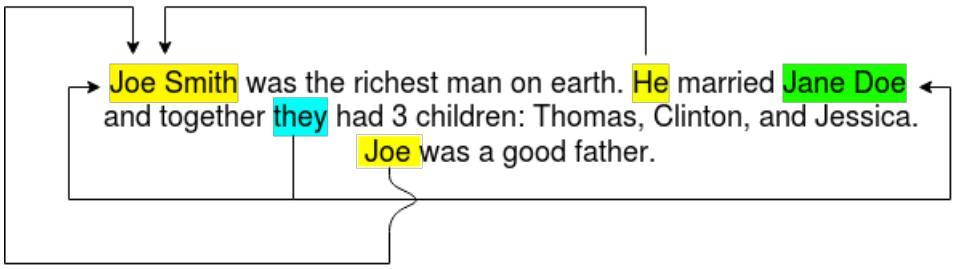


Figure 2.3: Anaphora Resolution Example

Using a naive approach can be mistaken for bringing the authenticity of Wikipedia free form text into question since most pronouns do refer to the article person. However, this does not detract from the overall problem because the point of the training data from Wikipedia is to capture the essence of being able to locate genealogical information. For this case, the most important anaphora to resolve are those that have genealogical info, and for now, it seems better to just use a naive approach since the pronouns either refer to the article person or a close by entity when talking about a related entity in English.

There are numerous other anaphora resolution systems as well, found at [17, 13, 29]. [17] is Stanford's implementation of the system discussed in [15], wrapped in a Python wrapper since it is native to Java. [13] is a Python wrapper for BART (also native to Java), which stands for Beautiful Anaphora Resolution Toolkit. This implementation utilizes a MaxEnt approach, and more can be found on their homepage [28]. [29] is CORT, a coreference resolution toolkit, that also has error analysis for its coreference results.

2.4.4 Machine Learning

The machine learning portion of this thesis utilizes *NLTK*, which stands for Natural Language Toolkit. This toolkit has multiple packages that are used in a variety of different places. For example, the *tokenize* package is used during preprocessing and the *classify* package is used when training the model. Specifically, the Naïve Bayes classifier is used from the *classify* package, which is one of many probabilistic machine learning models. The reason for choosing *NLTK* over the countless other libraries available is because it contains a lot of functionality that was needed in one place. It also has a plugin to use the *CoreNLP* library, which is native to Java.

The Naïve Bayes classifier is utilized in this thesis over other machine learning models because of its speed and efficiency in working with text data. This classifier is based on Bayes Theorem, shown in Figure 2.4. This theorem states that the probability of A given that B already occurred is equal to the probability of B given that A already occurred, multiplied by the probability of A occurring, divided by the probability of B occurring.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Figure 2.4: Bayes Theorem

Bayes Theorem is applicable to the Naïve Bayes classifier because it can be thought of as picking the class with the best probability of that class occurring given the document (represented by features) already occurring. With some assumptions and simplification, the formula shown in Figure 2.5 is reached.

One of the assumptions made is that each features conditional probability (P(x|c)) is independent of each other, meaning that the conditional for each probability is only

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c_j) \prod_{x \in X} P(x \mid c)$$

Figure 2.5: Naive Bayes Classifier

for the class, and not the other features. A bag of words assumption is also need, which assumes that the position of each feature is irrelevant. This gives the freedom for the individual conditional probabilities of each feature to be multiplied without constraint [22].

The Naïve Bayes classifier works really well in cases where the independence of features actually exists and the bag of words assumption can be taken. This thesis falls into this category because the features used are mainly in relation to the words that occur between the entities. The ordering of the words and the dependence on other words are not important because the goal is to be able to predict the relationships regardless of how they are said. Comparing to the Decision Tree classifier (another common model), the Naïve Bayes classifier is less prone to overfitting and can deal with fragmentation [22].

Chapter 3

RELATED WORK

3.1 Genealogy in General

Some academic work in genealogy is focused on bringing together separate sources of genealogical data under one common interface. [21] by Hansen is a work that reviews this process. Specifically it talks about how the Semantic Web and its foundational technology, Resource Description Framework (RDF), can be utilized to link data together and potentially realize the goal of one comprehensive family tree. The idea behind RDF is that it expresses data as properties in relation to URIs. This results in *subject* - *predicate* - *object* triplets. For example, *http://example.com/actors/JamesStewart* - *http://example.com/properties/birthDate* - *1908-05-20xsd:date* would be used to represent that James Stewart is born on May 20, 1908. The URIs would acts as the base for different properties. Hansen proposes a vocabulary which can be used to standardize the capturing of genealogical information through RDF triplets, with the hope that a standard will eventually allow for a global family tree.

3.2 Genealogy with Well-defined Data Sources

Many works in this field are devoted to genealogy with well-defined data sources. A few will be discussed below. As a general comment, the works in the particular domain of well-defined data sources are still crucial to the field of genealogy. There are many well-defined historical documents that need categorizing, and they can push that effort forward.

One such work is *OntoES*, an Ontology Extraction System [34], produced by Woodbury. At a high level, this work was oriented towards mapping genealogical information from parish and town records onto an ontology for quick access. An example of the type of data worked with is depicted in Figure 3.1.

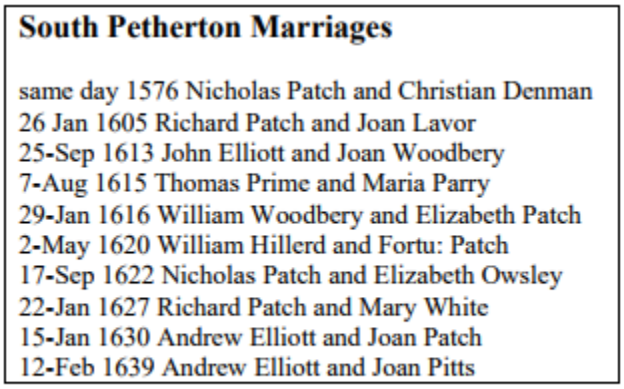


Figure 1. South Petherton marriages from GENUKI web page.

Figure 3.1: Data Example

The chosen methodology for matching information onto an ontology was the use of keyword recognizers, which might be more commonly referred to as regular expressions. For example, discovery of something that matches **(\b(md\.?|marry|marriage|married|maried|wed| |wedding)\b)** would signify the occurrence of a marriage. From this, two entities are inferred and an encapsulation of the data is mapped to the ontology. Once the information is extracted, SPARQL queries can be run to easily search the facts.

Another such work is *OntoSoar* [23], conducted by Lindes et al. The aim of their work is in the same vein: to automatically extract information from family history books and make that knowledge searchable. The specific area they focus in on is structured text that follows a formulaic style. This is common of most historical documents, which can have shorthand rules for quickly recording information for matters like marriages, births, deaths, etc. At its core, this system matches text onto

a user supplied ontology using lexical, syntactic, and semantic rules. In the pipeline, the text is divided into segments and each segment is then interpreted and analyzed to see whether any fact assertions can be overlaid to the ontology.

These works, among others like those found at [32, 20] have varying ranges of success. For example, the OntoES system achieved an F1-score of 0.979 in capturing marriages, while the OntoSoar system only achieved an F1-score of 0.714 in capturing marriages. These works do fall under the general domain of genealogy with well-defined data sources, but differences in the exact data sources can still account for large variability in success. One trend that can be seen through these works though is the usage of ontologies to map the genealogical information. Table 3.1 shows an aggregated view of the information about the two works described above.

Table 3.1: Aggregated View of Information for *OntoEs* and *OntoSoar*

Item	*OntoEs*	*OntoSoar*
Data Source	Parish and Town Records	Family History Books
Sample Size	967 Marriage records, 4505 Birth records, 4801 Death records	31 Persons, 14 Births, 9 Deaths, 7 Marriages, 16 Children
Types of Extracted Relationships	Spousal Relationships	Child to Parent Relationships, Spousal Relationships
Evaluation	Manual Verification	Manual Verification

3.3 Genealogy with Free Form Text

On the other hand, the area of genealogy with free form text (as defined by this thesis) is still a developing domain. One problem that seems to arise is that many projects have different definitions of free form text and/or what it means for the text to be unstructured and unpatterned. From the perspective of this thesis, the text that those works look at still seem to follow some general patterns. One giveaway is that the text they are analyzing is text that has been recorded for the sole purpose of documenting a fact (for example: church records, royalty documents). They are thus working in a slightly different domain than this thesis.

One such work that falls into the above criteria is [18], pushed forward by Efremova et al. Efremova et al. describes two techniques for determining relationships from notary acts. An example of a notary act is shown in Figure 3.2. The first technique is a classification approach that follows a similar pipeline to this thesis. From a text, it generates all possible entity pairs and tries to locate features that can help distinguish types of relationships. Feature vectors are then passed to a supervised machine learning model to train it. The second technique is a Hidden Markov Model approach, which annotates the text with tags and uses relationship descriptors to connect related entities. Overall, Efremova et al. examines a corpus of size 347, which is increased by a technique that generates more similar instances. The precision, recall, and f-score for the classification approach and Hidden Markov Model approach are reproduced below in Figure 3.3 and Figure 3.4 for convenience.

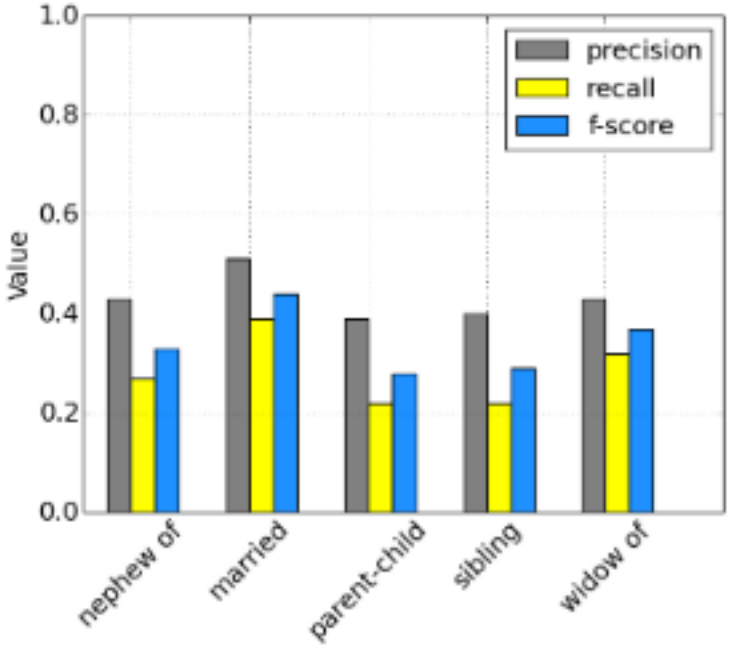
This thesis is different in that entity pairs are chosen more selectively, a wider range of features are tested, and a much larger and diverse corpus is used for training and testing. Additionally, preprocessing steps taken in this thesis use current state-of-the-art libraries that can be later substituted by better libraries for improved performance.

Dit document certificeert: Jan de Jager en **zijn vrouw** Hendrina Jacobs, verklaren afstand te doen van alle rechten van de akte van koop en verkoop van 02/10/1906, opgemaakt voor notaris van Breda, ten behoeve van Martinus van Doorn, winkelier te Uden.
*This document certifies: Jan de Jager and **his wife** Hendrina Jacobs, declare to waive all rights of the act of sale and purchase of 02/10/1906, registered at the notary Breda, with beneficiary Martinus van Doorn, shopkeeper in Uden.*

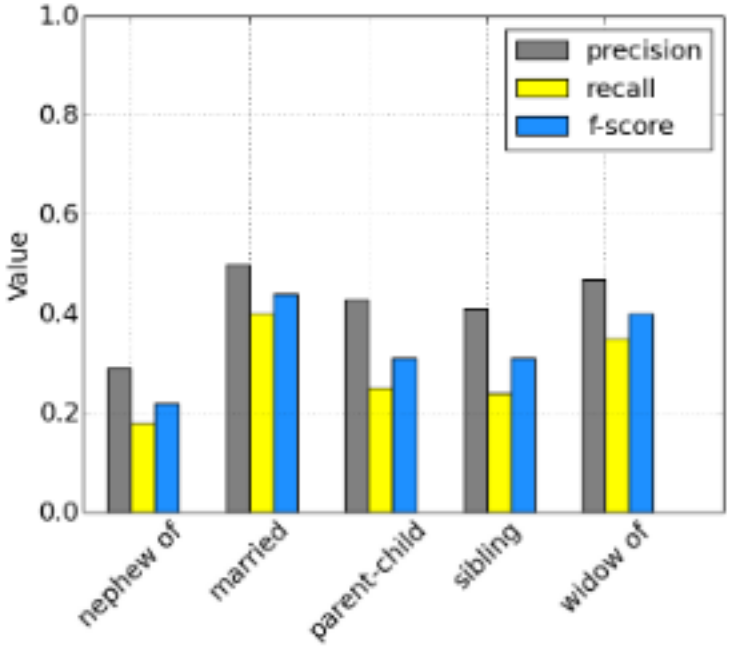
Figure 3.2: Notary Act

[19] is a work by Embley et al. that dives into genealogy extraction from obituaries. An example obituary for that work is shown in Figure 3.5. From the standpoint of this thesis, obituaries are close to free form text in that the text is unstructured, but there are general patterns that are specific to obituaries that give the text some patterns which can be exploited. For example, rules can made to map keywords to the type of relationship, like the presence of the word "brother" or "sister" to signify a sibling relationship. Domain knowledge could then be applied to create rules for inferring the actual entities. In addition, the style of obituaries guarantees the presence of genealogical relationships, which also puts the work in a slightly different domain. Examination of this work shows that an ontology approach with keyword recognizers is used, which achieves an F1-score of 0.865 when capturing relationships.

[16] is a work by Culotta et al. which also operates with Wikipedia as its source of data and similarly uses a model to predict relationships (although they use linear-chain conditional random fields). However, this work is different than this thesis in that it takes a narrower scope in regards to entity pairs and a broader scope in regards to capturing other pieces of information besides genealogical relationships. They take advantage of the fact that there is a main entity in a Wikipedia page and



(a) bi-grams of words and standard classification



(b) bi-grams and binary classification

Figure 3.3: Classification Approach Results

only look at entity pairs where one of the entities is the main entity. This consequently limits their system to predicting information for a single main entity. This thesis is different in that it tries to extract all genealogical relationships from the Wikipedia text, regardless of who the main entity is. The hope is that the system can be generalized enough to extend beyond Wikipedia (as discussed in Chapter 1), but to reiterate, there is still some dependency on Wikipedia since the model is trained only on Wikipedia. Culotta et al. are able to achieve an F1-score of 0.6136, although this value is for capturing any information rather than just genealogical information.

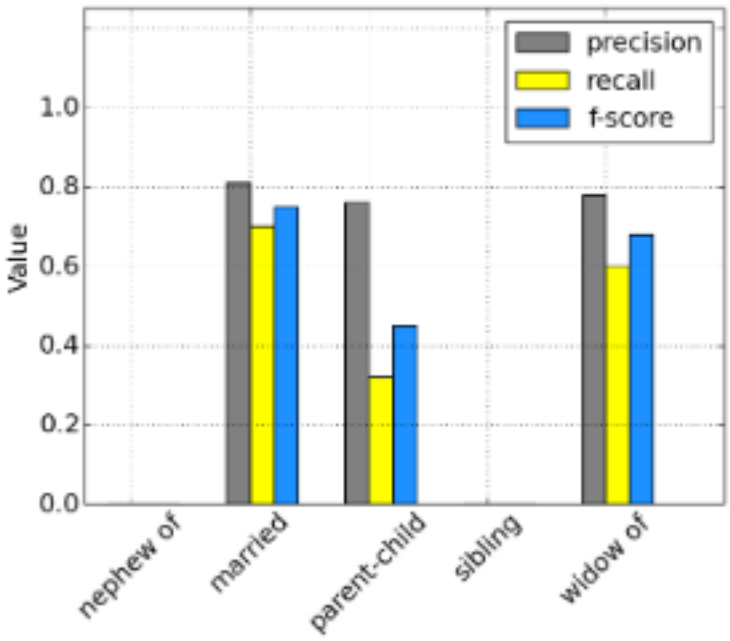
Table 3.2 shows an aggregated view of the information about the three works described above.

Table 3.2: Aggregated View of Information for Efremova et al, Embley et al., and Culotta et al.

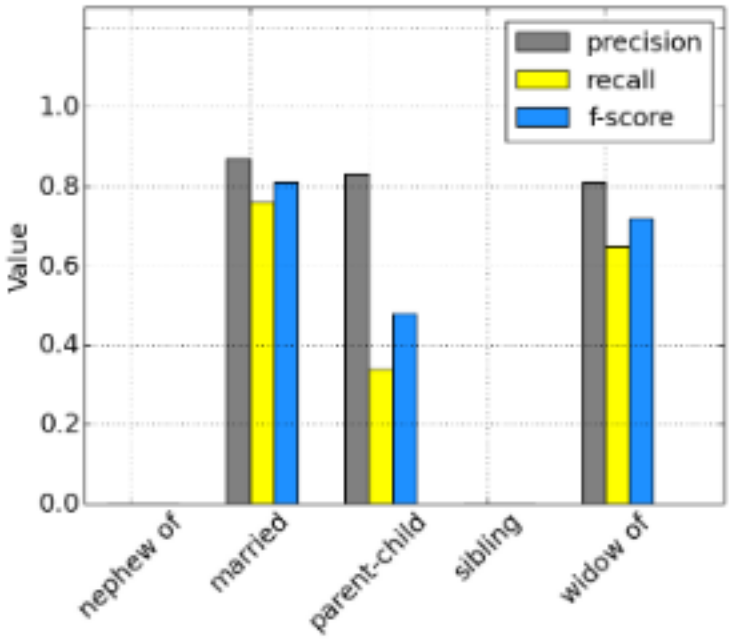
Item	Efremova et al	Embley et al.	Culotta et al.
Data Source	Notary Acts	Obituaries	Wikipedia
Sample Size	347 Notary Acts (1005 relations)	128 Obituaries (1079 relations)	1127 Paragraphs from 271 Articles
Types of Extracted Relationships	Nephew, Spousal Relationships, Child to Parent Relationships, Sibling Relationships, Widow	Child to Parent Relationships, Spousal Relationships, Sibling Relationships	Child to Parent Relationships, Spousal Relationships, Sibling Relationships
Evaluation	Manual Verification	Manual Verification	Manual Verification

As a whole, this thesis seems to stand out because of its decision to train and

test across a much larger and unprocessed sample. The trade off for doing this is an increased difficulty in achieving a fully labeled data set, but operating in these circumstances also provides a perspective on the performance of the chosen technique in an imperfect world. This can serve as the lower bound, given that there could be a more complete data set.



(a) using HMM model to annotate person names and relationship descriptors in a historical notary act



(b) using HMM to annotate relationship descriptors and pattern-based NER to annotate names

Figure 3.4: Hidden Markov Model Approach Results

Brian Fielding Frost

Our beloved Brian Fielding Frost, age 41, passed away Saturday morning, March 7, 1998, due to injuries sustained in an automobile accident. He was born August 4, 1956 in Salt Lake City, to Donald Fielding and Helen Glade Frost. He married Susan Fox on June 1, 1981.

He is survived by Susan; sons Jordan (9), Travis (8), Bryce (6); parents, three brothers, Donald Glade (Lynne), Kenneth Wesley (Ellen), Alex Reed, and two sisters, Anne (Dale) Elkins and Sally (Kent) Britton. A son, Michael Brian Frost, preceded him in death.

Funeral services will be held at 12 noon Friday, March 13, 1998 in the Howard Stake Center, 350 South 1600 East. Friends may call 5-7 p.m. Thursday at Wasatch Lawn Mortuary, 3401 S. Highland Drive, and at the Stake Center from 10:45-11:45 a.m. Friday. Interment at Wasatch Lawn Memorial Park.

Figure 3.5: Obituary Example

Chapter 4

SYSTEMS

This thesis proposes a novel three part system which can automatically interpret free form text to extract relationships and produce a family tree compliant with GEDCOM formatting. Each part of the system lines up with one of the contributions, listed in Chapter 1. They are reproduced below to provide a roadmap of what the immediate following sections will discuss:

1. A generic and extendable golden data set which can be utilized by other genealogists in their research on free form text.

2. A pipeline for converting free form text into feature vectors and training a Naïve Bayes machine learning model, all the way from text cleaning to named entity recognition and anaphora resolution.

3. A system for converting chosen relationships into a forest of family trees in the GEDCOM format.

4.1 Subsystem 1: Data Gathering

The first subsystem is in charge of creating and updating a generic and extendable database of training data from Wikipedia, the designated source of free form text. The end goal for this database is to hold records that state that entity A and entity B have some sort of relationship R that is found in free form text X. The overall pipeline of this subsystem, at the highest level of abstraction, is visually represented in Figure 4.1. A Wikipedia page is verified, entity information and relationship information is scraped from its corresponding Wikidata page, and relevant information,

like the Wikipedia page contents, the entities, and the relationships, are saved into the database. At the writing of this thesis, around 339248 Wikipedia pages, 232344 relationships, and 338140 entities were scraped.

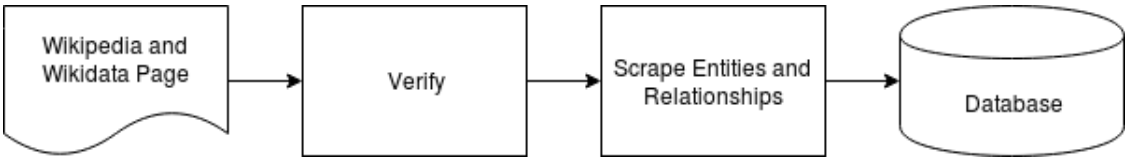


Figure 4.1: Data Gathering Pipeline

Additionally, this subsystem can be understood as a web scraper. It visits a Wikipedia page, verifies it, scrapes and saves its relevant information, and then visits all the Wikipedia pages that are linked on the current page to also scrape the relevant information from those pages. This process is repeated again and again to collect as much data as needed.

Before further explanation on this subsystem, it is important to briefly touch on the database design that backs the whole process. There are three main groups of collections:

1. Staging Collections
2. Progress Collections
3. Corpus Collections

The staging collections are used to keep track of whether Wikipedia pages were verified and saved. The progress collections were used to keep track of whether the Wikipedia pages (and their corresponding Wikidata pages) were expanded and scraped yet. The corpus collections were used to save the free form text from the Wikipedia page, the entities found on each page, and the relationships between different entities. This database design was used specifically to accommodate concurrency

while using MongoDB. The first attempt, which utilized a single web scraper, proved to be too slow and the design was shifted to a parallel approach.

The staging collections and the progress collections allowed for multiple web scrapers to work together to gather data without repeating work already done by another web scraper. Whenever a web scraper started verifying and saving a page, it wrote to the staging collections. Other web scrapers would then check the staging collections to see if the page they were about to process was already there. If it was, it knew that it could pick another page to process instead. Similarly for the progress collections, a web scraper would write to the progress collections when expanding links and scraping entity and relationship information from a page so that other web scrapers do not expand and scrape the same page.

Multiple bots are implemented to split this web scraping problem into smaller, manageable parts. The first bot, visually displayed in Figure 4.2, was in charge of pulling a link from the "To Stage" collection and verifying that the linked Wikipedia page was actually about a person. This was done by checking the corresponding Wikidata page and ensuring that the field for **INSTANCE OF** was set to **HUMAN**. If this verification passed, the page contents were saved into the database and the page became available for link exploration. After the bot finished, the link was moved from the "To Stage" collection to the "Staged Collection". In addition, a reference to the saved page contents was enqueued into the "To Expand" collection for the next step. The pseudocode is shown in Algorithm 1.

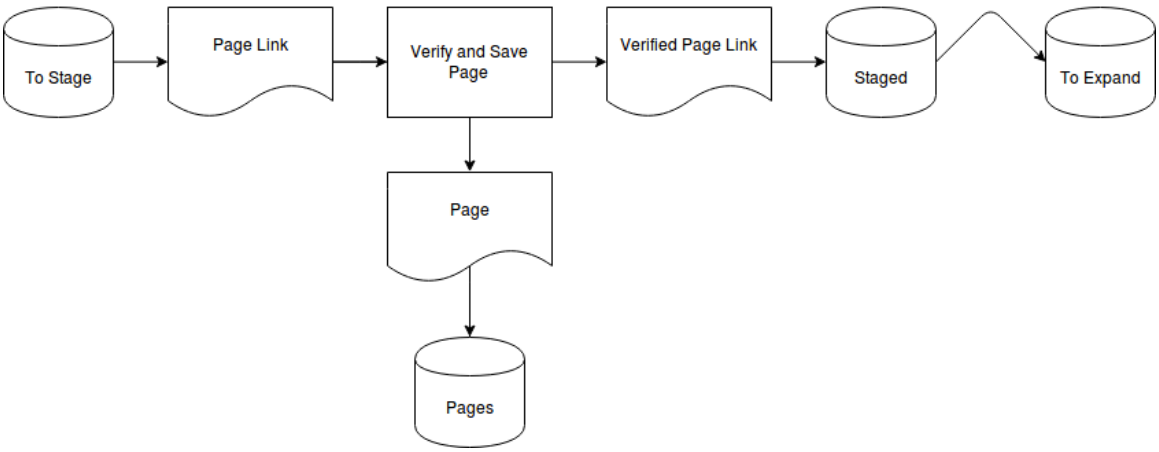


Figure 4.2: Verify and Save Bot

while *"To Stage" is not empty* **do**
    link_document := next document from "To Stage";
    if *link_document in "Staged"* **then**
        sleep 1 second;
    else
        move link_document from "To Stage" to "Staged";
        if *link_document INSTANCE OF == HUMAN* **then**
            insert page to "Pages";
            insert page_document to "To Expand";
        end
    end
end

Algorithm 1: Verify and Save Bot Pseudocode

The second bot, visually displayed in Figure 4.3, was responsible for dequeuing a page to explore from the "To Expand" collection and pushing all the links from that page into the "To Stage" collection for the bot described above to verify and save. Thus, there is a cycle here with the first bot that continually grows the database. This bot additionally pushes the page from the "To Expand" collection to the "Explored"

collection, so that the next bot can know when to start its own processing. The pseudocode is shown in Algorithm 2

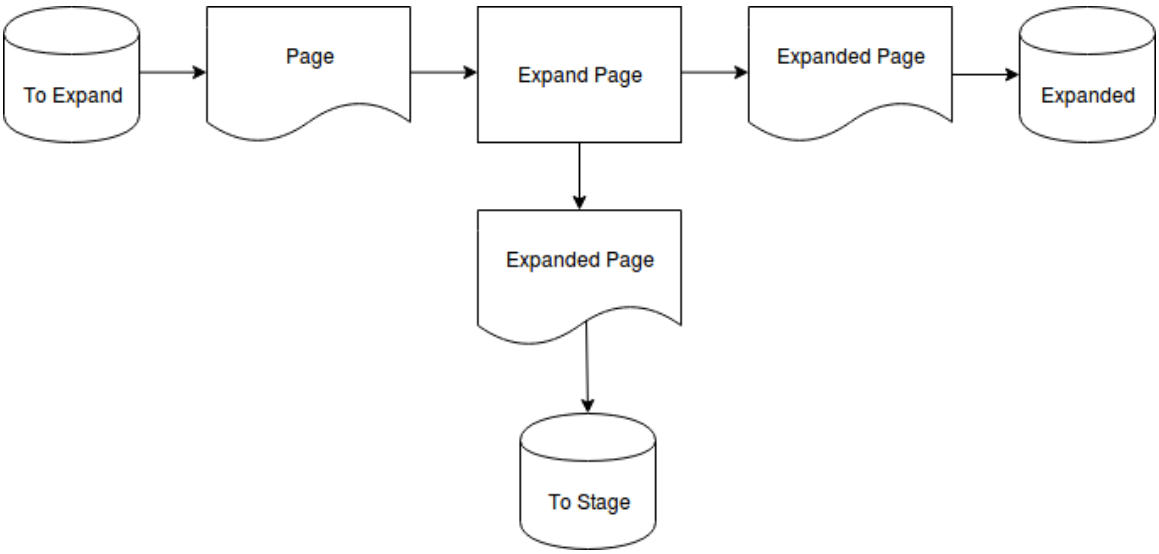


Figure 4.3: Expand Links Bot

while *"To Expand" is not empty* **do**
    page_document := next document from "To Expand";
    if *page_document in "Expanded"* **then**
        sleep 1 second;
    else
        move page_document from "To Expand" to "Expanded";
        for *each link in page_document* **do**
            insert link_document to "To Stage";
        end
    end
end

Algorithm 2: Expand Bot Pseudocode

The third bot, visually displayed in Figure 4.4, was set to pick a page to scrape from the "Expanded" collection. It then scraped the corresponding Wikidata page

to pick out information about the person that the Wikipedia article was about. In the next step, it looked at all the people listed as being related to the current person, and scraped the information for those people as well. The information about the different people were encapsulated and saved in the "Entities" collection, while the relationship information was saved in the "Corpus" collection. For each sibling and spousal relationship found during the scrape, the two different permutations of the two entities were inserted into the database. For example, if Bob and Jane were siblings found in Bob's Wikipedia and Wikidata page, two documents were inserted. One where Bob came first and another where Jane came first. For the child to parent relationship, only one document was inserted, where the first entity is the child and the second entity is the parent. All of this was done to preserve ordering, since the child to parent relationship is ordered. Lastly, the page was moved from the "Expanded" collection to the "Done" collection. The pseudocode is shown in Algorithm 3.

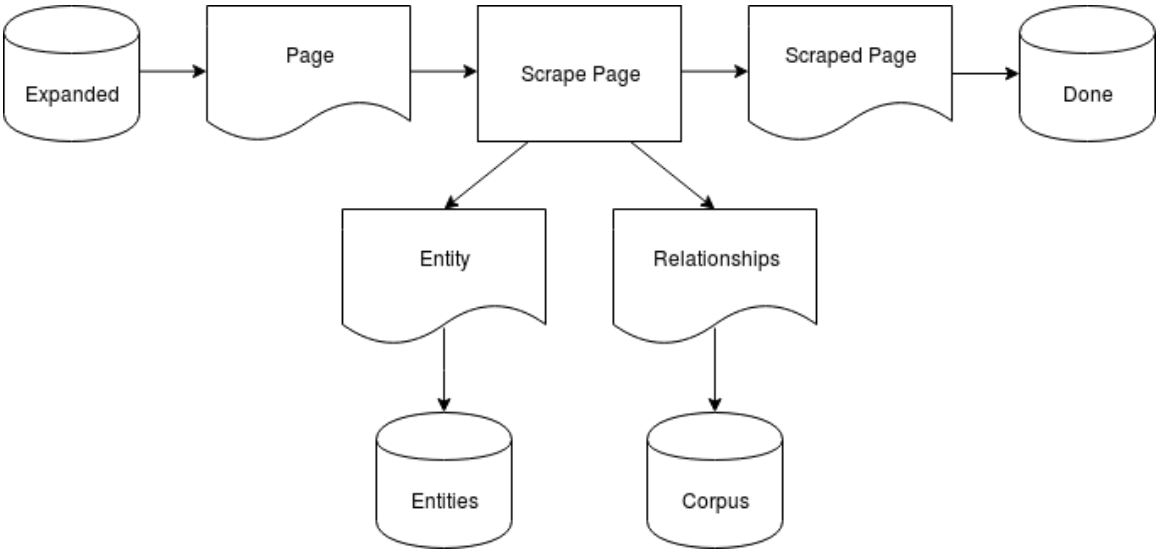


Figure 4.4: Scrape Entity and Relationship Information Bot

```
while "Expanded" is not empty do
    page_document := next document from "Expanded";
    if page_document in "Done" then
        sleep 1 second;
    else
        scrape entity information and insert to "Entities";
        scrape relationship information and nested entity information and
         insert to "Corpus" and "Entities", respectively;
        move page_document from "Expanded" to "Done";
    end
end
```

Algorithm 3: Scrape Bot Pseudocode

An interesting problem faced in the creation of these bots was ensuring that concurrency could be achieved without replicating work. As mentioned in Chapter 2, MongoDB lacks the capability of locking a document. This is a problem because a type of lock is needed so that multiple bots do not dequeue the same object to look at and process. To solve this problem, a primary key write-lock strategy was implemented. This trick relies on the fact that MongoDB uses a write lock on documents and so if multiple bots try to write to the same document, one necessarily has to go first. The primary key part comes into play because each record must have a unique primary key. When multiple bots try to write the same primary key into the collection, only one will ever succeed. Thus, the bot which successfully writes the primary key gets to process the dequeued object, while the losers have to dequeue another object and potentially contest against each other again.

In and of itself, this subsystem and its produced database can be considered an immense contribution to the field of genealogy. It provides a base upon which more extraction and data gathering can occur, as well as provides a corpus upon which other

genealogists and computer scientists can further research genealogy in the realm of free form text.

4.2 Subsystem 2: Natural Language Processing and Machine Learning

The second subsystem is responsible for building and training the Naïve Bayes model using the data generated by the first subsystem. The end goal for this model is to be able to predict whether two entities have a relationship specified in some free form text, and if there is, what type of relationship. Note that there is some filtering done on the data at this point. This is done in relation to the point talked about in Chapter 1 about how some Wikipedia pages may specify a relationship in the article which is not captured in the Wikidata page, and some Wikidata pages may specify relationships in some of its fields which are not represented in the Wikipedia page. This filtering is necessary because a known relationship that is specified in the Wikidata but is not represented in the Wikipedia page text cannot be expected to be found by the model, since the model is using the free form text to extract the relationships. This subsetting of the corpus and the methodologies used are further discussed in Chapter 5.

The input to this second subsystem is a series of free form texts (scraped by Subsystem 1 and containing relationships after the filtering), which are then processed and feature vectorized down to the relationship level with entity pairs. The feature vectors for each entity pair are then fed into a supervised learning classifier, which then learns the patterns associated with each type of relationship. The four classes that are currently supported are "Child-Parent", "Spouse", "Sibling", and "None". The idea behind only choosing these four classes was because capturing just those four relationships for every individual would theoretically allow for the combination of smaller trees into bigger forests. The overall diagram can be seen in Figure 4.5,

which is similar to other Machine Learning Model work flows.

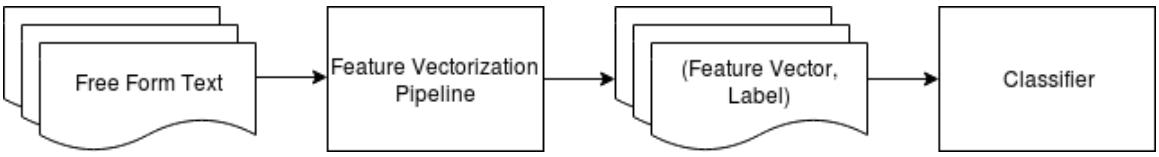


Figure 4.5: Machine Learning Work Flow

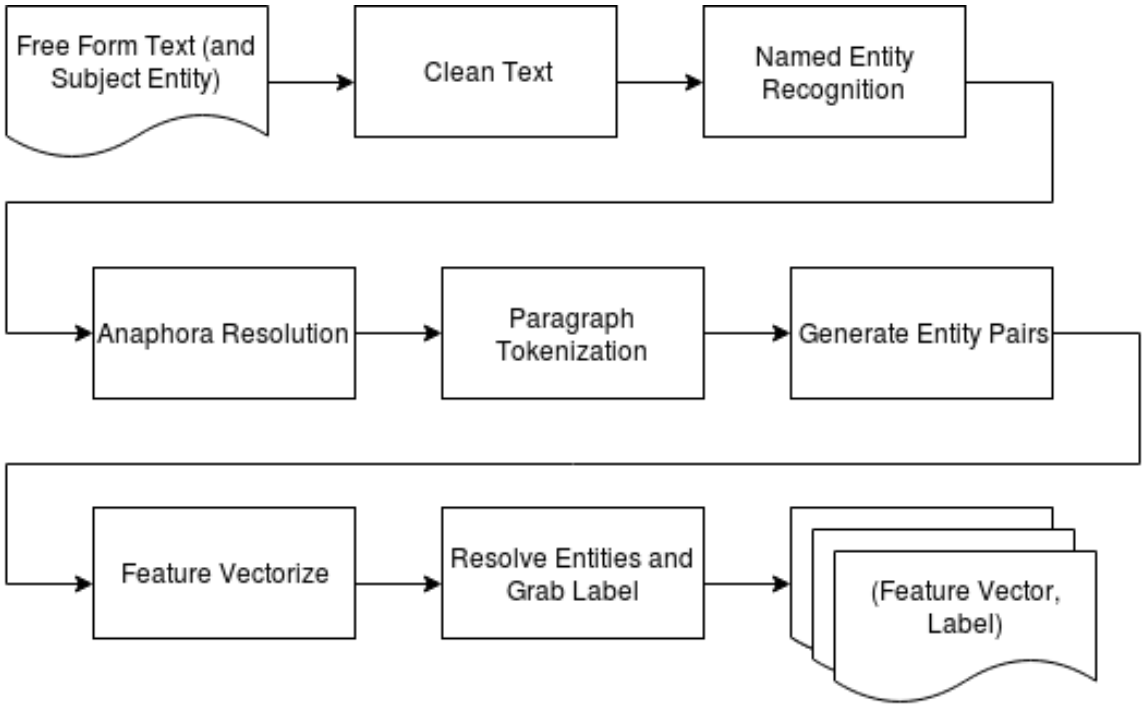


Figure 4.6: Feature Vectorization Pipeline

The feature vectorization pipeline has multiple steps to preprocess and convert free form text into feature vectors. See Figure 4.6 for the visual representation. The text is first cleaned to remove any text formatting that was carried over from the scraping. Named entity recognition is then performed to tag words with their corresponding entity tag. That combined data is passed to the anaphora resolution process to resolve pronouns to their proper nouns and to possibly connect proper nouns with other proper nouns. Trials for named entity recognition and anaphora resolution to document their performances are discussed in Chapter 5. After these

preprocessing steps are completed, the pipeline is ready to use the augmented free form text to generate entity pairs and their corresponding feature vectors, as well as assign a class label ("Child-Parent", "Spouse", "Sibling", or "None").

In order to generate the entity pairs, the text is split into paragraphs and only entities within the same paragraph are ever grouped together. This step limits the number of entity pairs to examine because it operates under the assumption that two entities must be in the same paragraph in order for them to "have" a genealogical relationship. Realistically, there are cases where this might not be true in the English language, but doing this step helps to eliminate excess entity pairs which most likely do not have a relationship. For example, an entity that is only found in paragraph 1 is most likely not related to an entity that is only found in paragraph 9.

Once the entity pairs are generated, the feature vectors can be built. As with every machine learning exercise, the chosen features are crucial in determining whether the end model is accurate in predicting new instances that were not encountered during training. Features have to be chosen that can help disambiguate the different classes. The difficulty at this point in this thesis was picking what text would be used for the feature vectorization. The same entity pair that was discovered in one paragraph could also exist in another paragraph. Additionally, multiple occurrences of the entity could happen in a paragraph. The problem thus boiled down to picking which paragraphs to look at, and which occurrences within those paragraphs to look at if there were multiple occurrences. The approach this thesis decided on was to look at the first paragraph with both entities and the paragraph with the shortest distance between the entities. The chosen occurrences and features are broken down for the two paragraphs in Table 4.1 and Table 4.2.

The words in between the entities, minus stop words, were chosen because they provided context to what was being discussed in between the two entities. If the

machine learning model saw that a specific word was constantly mentioned in between two entities when they actually had a relationship, it would know to associate that word with the relationship type. The possessive boolean feature was chosen because it could help to capture whether an entity was being discussed in relation to another entity. For example, the mother of entity A is usually presented as an entity B possessed by A. A feature for whether the entities occurred in the same sentence was also chosen because it is highly likely that a relationship is discussed in one sentence. Ratios of the entities appearances were also considered to account for the likelihood that an entity that is less discussed will be related to one who is more discussed. The last two features dealt with whether any other entities existed between the two entities in question. The hope was that this feature could help identify entities not in relationships, since it is likelier that when more entities are discussed in between, there is less chance of a relationship between the two entities.

Table 4.1: Chosen Features for First Paragraph with Both Entities

Paragraph Chosen	Occurrence Chosen	Feature
First paragraph with both entities	First occurrences of each entity	Words in between the entities
First paragraph with both entities	First occurrences of each entity	Whether entity 1 is possessive
First paragraph with both entities	First occurrences of each entity	Whether entity 2 is possessive
First paragraph with both entities	First occurrences of each entity	Whether entities are in the same sentence
First paragraph with both entities	First occurrences of each entity	Ratio of entity occurrences
First paragraph with both entities	First occurrences of each entity	Whether another entity is in between
First paragraph with both entities	First occurrences of each entity	Count of entities in between

Table 4.2: Chosen Features for Paragraph with Shortest Distance Between Entities

Paragraph Chosen	Occurrence Chosen	Feature
Paragraph with shortest distance between entities	Closest occurrences of each entity	Words in between the entities
Paragraph with shortest distance between entities	Closest occurrences of each entity	Whether entity 1 is possessive
Paragraph with shortest distance between entities	Closest occurrences of each entity	Whether entity 2 is possessive
Paragraph with shortest distance between entities	Closest occurrences of each entity	Whether entities are in the same sentence
Paragraph with shortest distance between entities	Closest occurrences of each entity	Ratio of entity occurrences
Paragraph with shortest distance between entities	Closest occurrences of each entity	Whether another entity is in between
Paragraph with shortest distance between entities	Closest occurrences of each entity	Count of entities in between

At this point, this subsystem utilizes the corpus produced by the first subsystem

(which was additionally filtered and subsetted at the beginning of this subsystem) to assign labels to each entity pair. To do this, the entities in the pairing are checked to see whether they are resolvable from the entities found during the scraping. This is slightly different than the subsetting problem because it is trying to resolve the scraped entities to the entities discovered through the named entity recognition, instead of just trying to locate the scraped entities in the text through string comparison. However, given the nature of the problem, a similar methodology can be applied. Trials to show the performance of the resolution of entities in the subset corpus to the entities discovered through the named entity recognition are documented in Chapter 5. If the entities are resolvable, and the entities are stated to have a relationship in the subset corpus, the type of relationship is assigned as the label. For all other entity pairs, the assigned relationship is the "None" case.

With all the feature vectors assembled and their labels assigned, the data can then be pushed into the machine learning model. The set up of the different Naïve Bayes classifiers tested against and their results are further discussed in Chapter 6.

Certain known issues are present as a result of the data used and the specific implementation. For example, an issue relates back to the problem of picking paragraphs to look at when building the feature vector. It is possible that the wrong paragraphs are chosen, and thus the model is training on a feature vector that is incorrectly paired with its class label. This problem arises despite the fact that the relationships are known after the scraping because the scraping portion does not give us any insight into where the actual relationships are stated.

In addition, genealogical relationships could have existed between two entities, but only the relationships in relation to the article person are recorded in the Wikidata. This means that some feature vectors might be incorrectly paired with the "None" label.

The named entity recognition, anaphora resolution, and resolution of entities from the subset corpus also compounds into an issue when assigning the label. It means that the "None" class is assigned more often than it should have been, either because the entity was incorrectly recognized, not resolved in the right places, or not resolved from the subset corpus. In terms of the model, this means that it trains on feature vectors with the "None" class even though they do have a relationship, and that some predictions for relationships are incorrectly verified as wrong.

Unfortunately, these are issues that have to be accepted as is based on current conditions. Unless there is a better data set that has been manually tagged, this thesis depends on a whole pipeline of processes that each have inherent error.

4.3 Subsystem 3: Family Tree Generation

The third and final subsystem is in charge of taking the relationships approved by the model and building the family trees in the GEDCOM format. In order to validate the outputted GEDCOM file, this thesis uses the GEDCOM validator found at [8], which quickly ensures that the file follows all guidelines for GEDCOM formatting, as well as provides some aggregated numbers.

The pipeline for this subsystem is shown in Figure 4.7. The input to this subsystem is a series of entity pairs and their relationship. Each entity is registered if it does not yet exist in the list of seen entities. Otherwise, it is connected to the existing entity. The relationship between the two entities is then registered into a family if a family exists for either entity. Otherwise, a new family is created between the two entities. For example, if a family already exists for the first entity with another entity, the second entity is added to that family. Alternatively, if a family exists for the second entity with another entity, the first entity is added to that family. Once all entity pairs and their relationship are processed, the final GEDCOM file is outputted, which

contains all the registered entities, all the registered families, and the additional meta data that was discussed in Chapter 2. A sample GEDCOM output file produced by this thesis for a random Wikipedia page can be found in Appendix 7.1.

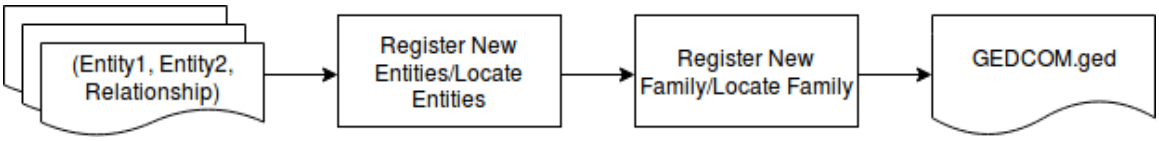


Figure 4.7: Subsystem 3 Pipeline

Chapter 5

EXPERIMENTAL DESIGN

This Chapter is dedicated to explaining how the smaller, individual pieces in Subsystem 2 play into the bigger machine learning exercise, which is discussed with its results in Chapter 6.

5.1 Subsetting the Corpus

The subsetting of the corpus took on a rule based approach to include the relationships that are in both the Wikipedia page and the Wikidata page and to exclude the relationships that only appear in the Wikidata page but not the Wikipedia page. The assumption taken is that if both entities that are found to have a relationship in the Wikidata are also present in the Wikipedia free form text, it must be that the relationship is discussed. As a reminder, each document in the corpus already included entity A, entity B, their relationship R, and the associated free form text X. The crux of this subsetting problem was to resolve entity A and entity B and ensure that they both actually appear in the free form text X. Since this process comes first in the pipeline, the goal was to achieve at least 90% accuracy, theoretically implying that 90% of entities in the Wikidata (with relationships) can be accounted for in the model.

The first subsetting algorithm used the rules found in Table 5.1. This algorithm used full string matching in both the page links and the page text to make its prediction on whether that entity existed in the free form text. The rationale behind using the links was that most entities in Wikipedia pages are hyperlinked to their own pages. Consequently, a link was thought to be a good indication of whether the

entities existed in the page. The algorithm checked both Unicode and ASCII strings to account for differences that human contributors might not have correctly translated for. It also decided to look at shortened versions of the name in the free form text if the full versions could not be found (either Unicode or ASCII) in the links or the free form text. The rules for shortening the name in this experiment are found in Table 5.2. These rules tried to isolate first and last names.

Table 5.1: Experiment 1 Rules for Resolving Entity

#	Return Value	Rule
1	TRUE	if any of entity's names (in Unicode, lowered and stripped) are in the page links
2	TRUE	if any of entity's names (in Unicode, lowered and stripped) are in the text
3	TRUE	if any of entity's names (in ASCII, lowered and stripped) are in the pages links
4	TRUE	if any of entity's names (in ASCII, lowered and stripped) are in the text
5	TRUE	if any of entity's short handed names (in ASCII, lowered and stripped) are in the text
6	FALSE	if none of the above are met

Table 5.2: Experiment 1 Rules for Shortening Name

#	Rule	Before	After
1	if name includes ' of ', return name up to ' of '	Foaad of Calpoly	Foaad
2	if name is three words and first word is a royalty title (King, Queen, Prince, Queen, etc), return last two words	King Henry Charles	Henry Charles
3	if name is three words, return first and last words	Michelle LaVaughn Robinson	Michelle Robinson
4	if name is greater than three words and last word is a Junior or Senior suffix, return first word and last two words	Louise Keith Summers Jr	Louse Summers Jr
5	if name is greater than three words and first word is a royalty title, return first two words and last word	King Henry Arnold Charles	King Henry Charles
6	if name is greater than three words, return first and last words	Raphael Nadar Jose Reyes	Raphael Reyes

With these rules, this first subsetting algorithm achieved an overall accuracy of 82% over five trials of ten entities from different Wikipedia pages each using random sampling. This accuracy represents the amount of times the algorithm successfully

predicted that an entity was in the text when it actually was in the text, and the amount of times the algorithm successfully predicted that an entity was not in the text when it actually was not in the text. See Table 5.3 for the total counts and Table 5.4 for the results.

Table 5.3: Experiment 1 Counts

	actual TRUE	actual FALSE
predicted TRUE	33	4
predicted FALSE	5	8

Table 5.4: Experiment 1 Measurements

Accuracy	.82
TRUE precision	0.8918918919
TRUE recall	0.8684210526
FALSE precision	0.6666666667
FALSE recall	0.6153846154

By looking at the individual errors in the first subsetting algorithm, the second subsetting algorithm removed unnecessary rules and added ones that were clearly missing. The rules can be found in Table 5.5 and the rules for shortening the name can be found in Table 5.6. The biggest changes for the rules were that links were no longer to be checked against and that if the entity's first name could be found and no other entities from that page had the same first name, then the algorithm returned true. The reasoning for these changes was because the links often contained the entity name while the free form text did not, and because many references to entities actually just use the first name. The biggest change for the shortening of

names rules was that each part in two part names (those with a comma in the middle and not pertaining to Junior or Senior suffixes) were checked.

Table 5.5: Experiment 1 Rules for Resolving Entity

#	Return Value	Rule
1	TRUE	if any of entity's names (in Unicode, lowered and stripped) are in the text
2	TRUE	if any of entity's names (in ASCII, lowered and stripped) are in the text
3	TRUE	if any of entity's short handed names (in ASCII, lowered and stripped) are in the text
4	TRUE	if any of the entity's first names (in Unicode, lowered, and stripped) are in the text and only if no other entities from that page share a same first name
5	FALSE	if none of the above are met

Table 5.6: Experiment 1 Rules for Shortening Name

#	Rule	Before	After
1	if name includes ',', and is two parts and does not include Junior or Senior, return first part and second part	Queen Yeonji, the Beautiful Warrior	[Queen Yeonji, the Beautiful Warrior]
2	if name includes ' of ', return name up to ' of '	Foaad of Calpoly	Foaad
3	if name is three words and first word is a royalty title (King, Queen, Prince, Queen, etc), return last two words	King Henry Charles	Henry Charles
4	if name is three words, return first and last words	Michelle LaVaughn Robinson	Michelle Robinson
5	if name is greater than three words and last word is a Junior or Senior suffix, return first word and last two words	Louise Keith Summers Jr	Louse Summers Jr
6	if name is greater than three words and first word is a royalty title, return first two words and last word	King Henry Arnold Charles	King Henry Charles
7	if name is greater than three words, return first and last words	Raphael Nadar Jose Reyes	Raphael Reyes

With these new rules, this second subsetting algorithm achieved an overall accuracy of 94% over ten trials of ten instances each using random sampling. See Table 5.7 for the total counts and Table 5.8 for the results.

Table 5.7: Experiment 2 Counts

	actual TRUE	actual FALSE
predicted TRUE	75	6
predicted FALSE	0	19

Table 5.8: Experiment 2 Measurements

Accuracy	.94
TRUE precision	0.9259259259
TRUE recall	1.0
FALSE precision	1.0
FALSE recall	0.76

The main difficulties faced by this second subsetting algorithm is that it performed badly when people with the same name were in the same article, when people had names that are common English words (for example: Will), and when people changed their name. However, this rule based algorithm was able to pass the initial set threshold of 90% and work continued forward after this iteration.

5.2 Named Entity Recognition

Although the *CoreNLP* system was found to be very successful on the CoNLL 2003 dataset, a quick trial was also run to get an estimate of its performance when tagging people entities from the Wikipedia data used by this thesis. As this thesis was not

focused on improving named entity recognition, the results were taken as is. With a small random sample of five Wikipedia pages, the *CoreNLP* library was 82% accurate in correctly tagging a person entity with a person tag. There were some difficulties in picking up single name references, especially when the names were not of English origin. Occasionally, a location was tagged as a person.

Notice that this does not mean that only 82% of entities in relationships are captured by the thesis. This is due to the fact that a person entity is usually mentioned many times without any reference to any genealogical information. Thus, where named entity recognition really matters for this thesis is tagging the person entities when there is an actual genealogical relationship being discussed. The value for this was gathered in the same trials and it was found that the *CoreNLP* library was 88% accurate in correctly tagging a person ntity with a person tag when that entity was was in the context of an actual genealogical relationship.

5.3 Anaphora Resolution

The anaphora resolution utilized in this thesis went for a naive approach to resolving any anaphora. Whenever a pronoun occurs that has the same gender as the free form text's main subject, that pronoun is replaced with the article person. Otherwise, the pronoun is replaced with the last seen entity of the other gender. A simple gender classifier with around 85% accuracy was incorporated, but an option was included that could also pull the gender from the Wikidata. A quick trial with a small random sample of ten Wikipedia pages was run to document this number. Overall, this system was able to achieve 95% accuracy in correctly resolving an anaphora from the Wikipedia free form text to its correct proper noun (when the gender was pulled from the Wikidata). The result here also follows the same line of reasoning above in that where anaphora resolution (and its inner gender classifier) really matters

is resolving anaphora (and gender) when there is an actual genealogical information being discussed. Again, this value was also gathering during the trials. This approach was found to be only 90% accurate at correctly resolving an anaphora from the Wikipedia free form text to its correct proper noun (when the gender was pulled from the Wikidata) in the context of an actual genealogical relationship. Unlike in the named entity recognition case, this value is lower than the overall number, meaning it is less performant when it counts. This could be a result of the dependence on named entity recognition, and assuming perfect named entity recognition would theoretically raise this value.

5.4 Resolving Entities from Subset Corpus to Entities from Named Entity Recognition

This experiment provided an estimate of how many entities that were in the subset corpus were resolvable to entities discovered through named entity recognition. An average of 85% was obtained over ten trials of 4000 pages each using random sampling. Additionally, how many relationships that are in the subset corpus that were accounted for in the model was also calculated at this time. Over the ten trials, an average of 75% was achieved. Combined, those numbers gives a picture of the room for improvement, as well as the error in providing the label for the feature vector.

Chapter 6

RESULTS

This chapter will explain the tested configurations of the machine learning model devised in subsystem 2 and their consequent results. Both a Naïve Bayes multiclass classifier and individual Naïve Bayes binary classifiers for the relationship classes ("Child-Parent", "Spouse", and "Sibling") were tested. Each trial was run by random sampling 4000 pages from the subsetted corpus and balancing the classes to have equal numbers in the training set. This resulted in the model training on around 20000 feature vectors for each trial. The rationale behind balancing the training set was because more than 90% of the feature vectors were associated with the "None" class, and balancing would negate the effect of majority rules when training the model. The results displayed were an aggregation over 5 trials.

The two configurable parameters for each trial were whether to look up the gender during the anaphora resolution step or to use the gender classifier and whether to also balance the classes in the test set for the model. Choosing to look up the gender or to predict the gender was included because the used anaphora resolution system requires a gender. For the balancing of the multiclass classifier, the classes "Child-parent", "Spouse", and "Sibling" were balanced to have equal number of instances, while the class *None* was balanced to be equal to the number of instances for all relationships. For the balancing of the binary classifiers, all other classes (besides the main class the binary classifier was built for) were balanced to have equal number of instances, while the main class was balanced to be equal to the number of instances for the other classes case. Balanced and unbalanced test cases were tried to consider both ends of the spectrum in terms of reality. In the real world, an unbalanced number of classes is likely to occur (especially for the "None" class). However, testing the model with

a balanced number of classes also gives a clue as to how effective the chosen features are.

6.1 Multiclass Classifier

A multiclass model was trained that was capable of predicting all four classes ("Child-Parent", "Spouse", "Sibling", or "None"). This model achieved an average accuracy of 53.61% when looking up the gender and when balancing the test set, an average accuracy of 43.21% when looking up the gender and not balancing the test set, an average accuracy of 54% when using the gender classifier and balancing the test set, and an average accuracy of 44.52% when using the gender classifier and not balancing the test set. These accuracies are organized in Table 6.1. The percentage of instances successfully predicted to exist for each class are also found there. For example, a class that occurs twice, but is only successfully predicted once will have a capture rate of 0.5000.

Table 6.1: Multiclass Classifier Average Numbers

Parameters	Accuracy	"Child Parent" Captured	"Spouse" Captured	"Sibling" Capture	"None" Captured
Look up gender Balanced test	0.5361	0.3393	0.7625	0.8592	0.4199
Look up gender Unbalanced test	0.4321	0.3408	0.7694	0.8455	0.4274
Classify gender Balanced test	0.5400	0.3299	0.7649	0.8538	0.4278
Classify gender Unbalanced test	0.4452	0.3331	0.7301	0.8651	0.4410

6.2 Child to Parent Relationship Classifier

A binary classifier was trained for the *Child-Parent* relationship. This model achieved an average accuracy of 69.13% when looking up the gender and when balancing the classes, an average accuracy of 61% when looking up the gender and not balancing the classes, an average accuracy of 68.69% when using the gender classifier and balancing the classes, and an average accuracy of 60.01% when using the gender classifier and not balancing the classes. See Table 6.2 for an organized view of the average accuracies, in addition to the precision, recall, and F1 scores for prediction of the relationship. Aggregated values comparing the different binary classifiers for the different configurations are found at Figure 6.1, 6.2, 6.3, and 6.4.

Table 6.2: *Child-Parent* Classifier Average Accuracies

Parameters	Accuracy	Precision	Recall	F1
Look up gender Balanced test	0.6913	0.7020	0.6603	0.6811
Look up gender Unbalanced test	0.6100	0.0176	0.8847	0.0344
Classify gender Balanced test	0.6869	0.7099	0.6364	0.6698
Classify gender Unbalanced test	0.6001	0.0169	0.8918	0.0335

6.3 Spousal Relationship Classifier

A binary classifier was trained for the *Spouse* relationship. This model achieved an average accuracy of 75.18% when looking up the gender and when balancing the classes, an average accuracy of 48.90% when looking up the gender and not balancing the classes, an average accuracy of 75.04% when using the gender classifier and balancing the classes, and an average accuracy of 49.98% when using the gender classifier and not balancing the classes. See Table 6.3 for an organized view of the average accuracies, in addition to the precision, recall, and F1 scores for prediction of the relationship. Aggregated values comparing the different binary classifiers for the different configurations are found at Figure 6.1, 6.2, 6.3, and 6.4.

Table 6.3: *Spouse* Classifier Average Accuracies

Parameters	Average Accuracy	Precision	Recall	F1
Look up gender Balanced test	0.7518	0.6930	0.9034	0.7842
Look up gender Unbalanced test	0.4890	0.0153	0.9681	0.0301
Classify gender Balanced test	0.7504	0.6993	0.8817	0.7799
Classify gender Unbalanced test	0.4998	0.0152	0.9715	0.0300

6.4 Sibling Relationship Classifier

A binary classifier was trained for the *Sibling* relationship. This model achieved an average accuracy of 72.52% when looking up the gender and when balancing the classes, an average accuracy of 55.21% when looking up the gender and not balancing the classes, an average accuracy of 73.27% when using the gender classifier and balancing the classes, and an average accuracy of 53.15% when using the gender classifier and not balancing the classes. See Table 6.4 for an organized view of the average accuracies, in addition to the precision, recall, and F1 scores for prediction of the relationship. Aggregated values comparing the different binary classifiers for the different configurations are found at Figure 6.1, 6.2, 6.3, and 6.4.

Table 6.4: *Sibling* Classifier Average Accuracies

Parameters	Average Accuracy	Precision	Recall	F1
Look up gender Balanced test	0.7252	0.6629	0.9216	0.7710
Look up gender Unbalanced test	0.5521	0.0131	0.9599	0.0258
Classify gender Balanced test	0.7327	0.6715	0.9141	0.7741
Classify gender Unbalanced test	0.5315	0.0123	0.9587	0.0243

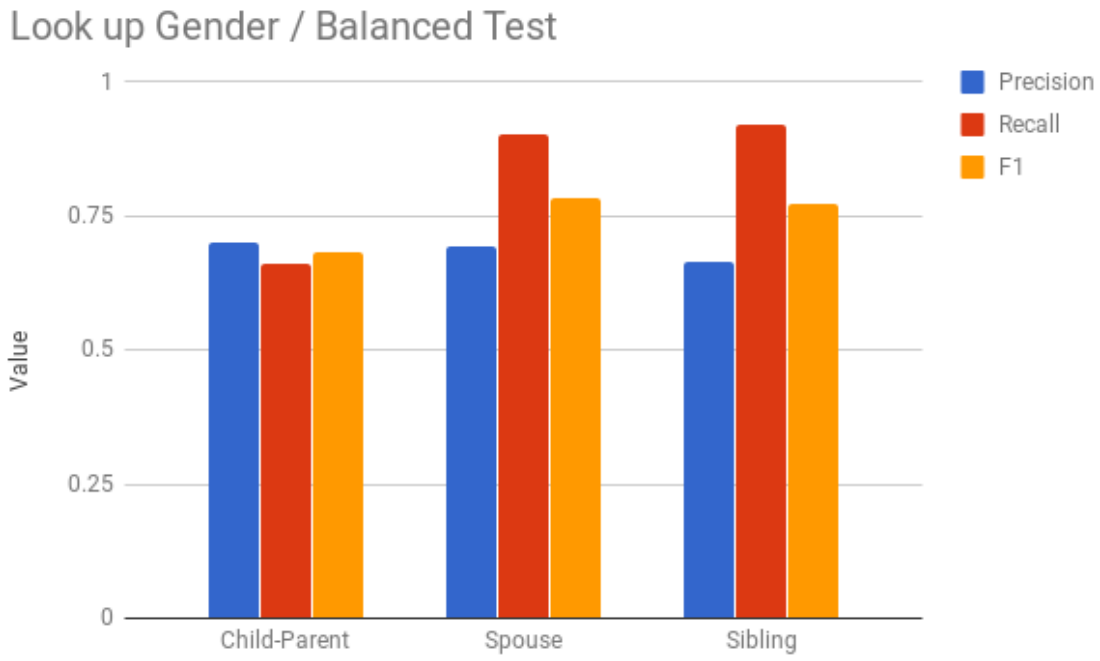


Figure 6.1: Scores when Looking up Gender and Balanced Test Set

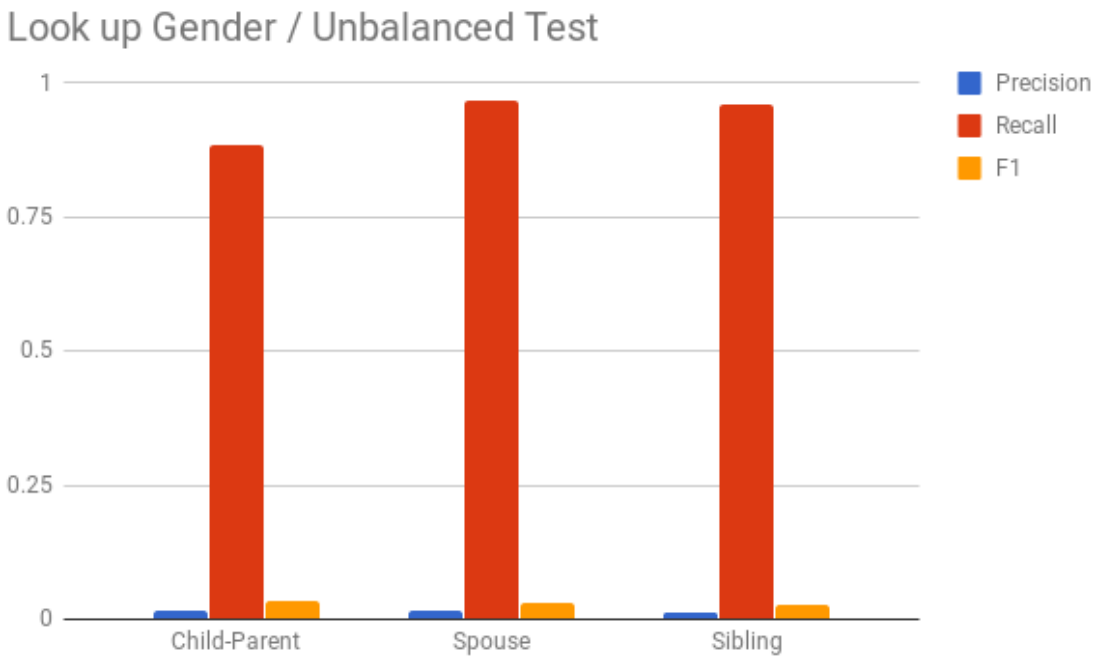


Figure 6.2: Scores when Looking up Gender and Unbalanced Test Set

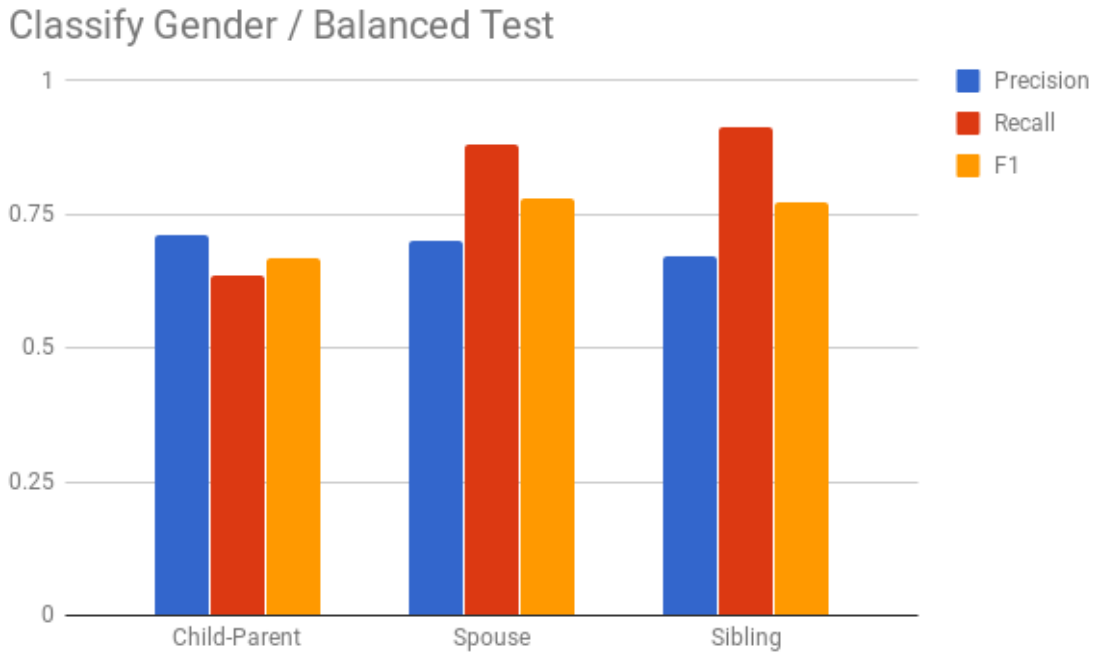


Figure 6.3: Scores when Classifying Gender and Balanced Test Set

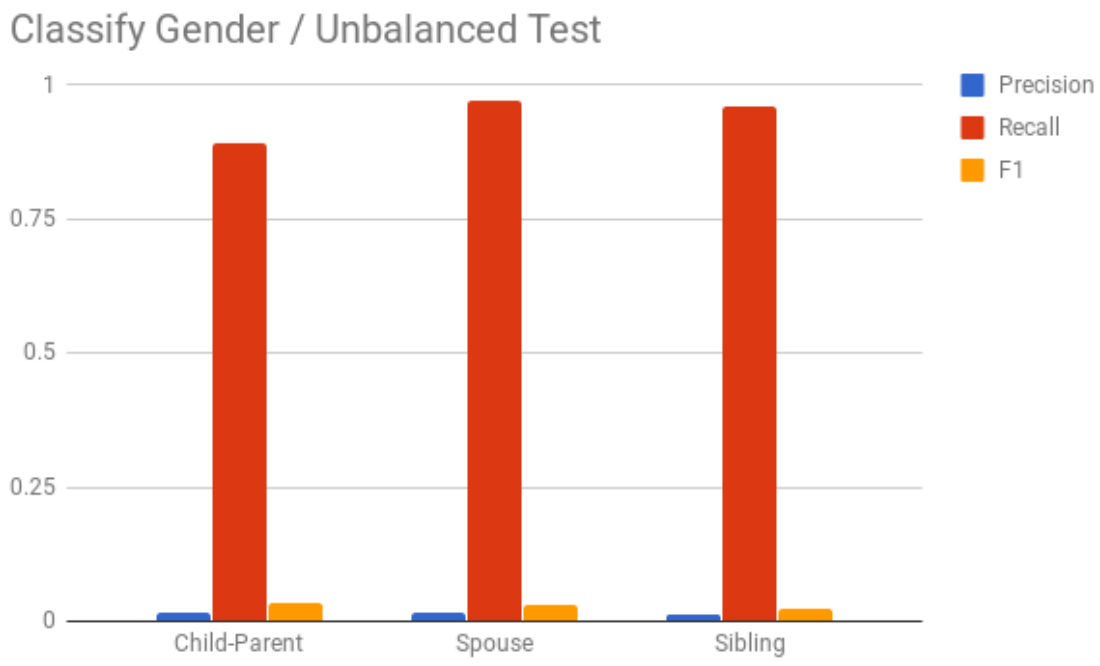


Figure 6.4: Scores when Classifying Gender and Unbalanced Test Set

6.5 Discussion of Classifier Performance

Overall, the results achieved are pretty impressive. The multiclass classifier had a best accuracy of 54% and the binary classifiers had best accuracies of 69% for the child to parent relationship classifier, 75% for the spousal relationship classifier, and 73% for the sibling relationship classifier. These numbers are despite the known issues discussed in Chapter 4, which negatively affected performance in two ways. First, because they were training the model with an incorrect label for the feature vector. Second, because they were providing incorrect feedback as to the results of some predictions.

For the multiclass classifier, although the percentages of cases successfully captured for each class were similar for each configuration, the accuracy is considerably less when using the unbalanced test set because the majority of the test cases were

the "None" class, which the model only captured around 42% of the time, on average. The intuition behind why the capture rates are similar, regardless of using a balanced or unbalanced test is because the model was trained in either case with a balanced training set. In considering why looking up the gender or classifying the gender has no bearing, remember the point discussed in Chapter 5: where anaphora resolution (and its inner gender classifier) really matters is resolving anaphora (and gender) when there is an actual genealogical information being discussed. It is likely that anaphora resolution was able to successfully resolve anaphora for the appropriate instances or that the locations of genealogical information did not need anaphora resolution in the first place. Taking a comparison standpoint, this multiclass classifier does considerably better than the base case random choice algorithm, which would have achieved 25% given that there are four classes.

Examining the binary classifiers, the same trend occurs where the accuracy dips when using the unbalanced test. What is interesting to see though is that the precision becomes really low and the recall becomes really high. The low precision is directly correlated with there being so many "None" cases and the inaccuracies of the classifiers in classifying those cases as an actual relationships rather than as "None". The high recall is potentially a result of the balancing technique, which balances after splitting the corpus into the train and test sets instead of before for the case where the test case is unbalanced. This results in a higher of relationship instances in the train, which then has more data to learn from. Looking at the accuracies of each classifier also signals that these classifiers do better than the base case random choice algorithm, which would have achieved 50% this time, given two classes.

Across the binary classifiers, the Child-Parent classifier seems to perform the worst. This is probably attributable to the fact that this specific relationship considers the direction of the relationship, while spousal and sibling relationships do not. The spouse and sibling classifiers do comparatively well, and this could possibly be

attributed to the fact that the entities in these two kinds of relationships are often stated closely together. This ordering of efficiency stays true for the multiclass classifier as well, where the child to parent relationships are captured least successfully, and the spousal and sibling relationships are captured successfully at a closer rate.

Theoretically speaking, it is possible to estimate how successful this system could be in capturing all the actual relationships between entities from the free form text. The values for the classifiers described above are subtly different than this because they measure the success of capturing relationships between entities given that the label is already assigned. Because of the smaller pieces in the system described in Chapter 5, there are some errors in assigning labels, and consequently, errors in capturing the actual relationships. Figure 6.5 shows the pipeline which leads up to the assignment of the label, along with the individual performance at each stage. The values for the named entity recognition and anaphora resolution use the overall performance to show the generic case. Multiplying all these together yields 62.24%, which is the theoretical value of how many labels were correctly assigned for the case where a relationship actually exists. This could then be multiplied with the accuracy of each classifier to determine the percentage of actual relationships that were classified correctly. For example, the Spousal relationship classifier would have theoretically captured 46.68% of the actual relationships.

Realistically, this number will actually be higher though because where the performance of the named entity recognition and anaphora resolution really matter is when they are performed on entities and anaphora that are mentioned when discussing a genealogical relationship. The quick estimates discussed in Chapter 5 showed the results to be better in the named entity recognition case, which would compound into a higher value in the overall pipeline. To take this one step further and assume perfect named entity recognition and anaphora resolution performance, the theoretical value of how many labels were correctly assigned for the case where a relationship actually

exists would be 79.9%. Realistically, this would also be higher, since the performance of resolving subset entities to entities from named entity recognition is dependent on the named entity recognition and anaphora resolution steps as well.

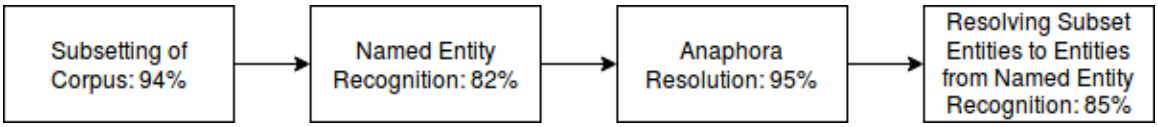


Figure 6.5: Theoretical Success

It is also beneficial at this point to compare this thesis to the work that is most similar. The work in [18], discussed in Chapter 3, is chosen for this task. Comparison to [16], the work that also deals with Wikipedia, is deferred because they do not provide results specific to genealogical relationships. Recall that the data source for [18] is different, as well as their definitions, so this is not an apples to apples comparison. In addition, the performance numbers for the classifiers have a slightly different meaning because they used assigned labels versus actual labels. Comparing to the classification technique, this thesis significantly outperforms their similar classification approach. In that case, child to parent relationships achieved a F1 score of around 0.3 (compared to 0.68 for this thesis), spousal relationships achieved a F1 score of around 0.45 (compared to 0.78 for this thesis), and sibling relationships achieved a F1 score of around 0.35 (compared to 0.77 for this thesis). Comparing to the Hidden Markov Model technique, this thesis did just as well for the spousal relationship, and better for the child to parent relationship. Note that the sibling relationship was not tested for in [18] for the Hidden Markov Model technique. In their case, they achieved a F1 score of around 0.8 for their spousal relationship and a F1 score of around 0.5 for their child to parent relationship.

Table 6.5 shows an aggregated view of the information about this work, similar to the ones found in Chapter 3.

Table 6.5: Aggregated View of Information for this Thesis

Item	Thesis
Data Source	Wikipedia
Sample Size	4000 Wikipedia Articles (around 20000 relations)
Types of Extracted Relationships	Child to Parent Relationships, Spousal Relationships, Sibling Relationships, None Relationships
Evaluation	Verification through Wikidata

Chapter 7

CONCLUSION

The primary goal of this thesis was to create a system which can automatically interpret free form text to extract relationships and produce a family tree compliant with GEDCOM formatting. This was achieved in three main steps. First, an extendable database of genealogical relationships were systematically scraped from Wikipedia free form text. Second, a machine learning model capable of predicting relationships was built and trained using the corpus collected in the first step. Third, a subsystem was implemented to accumulate the relationships chosen by the model into the GEDCOM format.

The overall results of the system show promise, especially as something that is trying to uncover new grounds in this defined domain of free form text. The results will also only get better as named entity recognition and anaphora resolution get better, so there is much to look forward to in the future. One problem, however, of conducting research in a relatively new domain is that there is not much work to effectively compare against. This thesis is compared to the closest work discovered, but it is not an apples to apples comparison.

Additionally, some issues arose due to the data source chosen for this thesis. Limitations in the data source in capturing all relationships and the location in the text where those relationships are explicitly stated meant that some assumptions had to be made. The assumptions tried to capture as many cases as possible, but there can definitely be edge cases.

7.1 Future Work

With regards to the implementation of this thesis, there are several aspects which can be improved. First, generalization of the system to better handle all cases of names, instead of depending on the English convention of first name, last name. Second, insertion of a better named entity recognition system when it becomes available. Third, improvement to the anaphora resolution system and its gender classifier (or insertion of a better anaphora resolution system when it becomes available). Fourth, enhancement of the process for resolving subset entities to entities discovered through named entity recognition. Fifth, better verification of the systems ability to capture all relationships versus assigned relationships through more manual tagging. And last, the addition of newer and better features which can help to better distinguish the different classes.

Other machine learning models could also be applied to this problem. While the Naïve Bayes classifier is a great choice, there could also be other systems that, when tuned, could perform better. This is a consideration that is present in all machine learning problems and only future research can tell.

BIBLIOGRAPHY

[1] 23andme. https://www.23andme.com/.

[2] Ancestry. https://www.ancestry.com/.

[3] Cal Poly Github. http://www.github.com/CalPoly.

[4] Dbpedia. http://wiki.dbpedia.org/.

[5] Family search. https://www.familysearch.org/.

[6] Family search: Let's put the world's historical records online in one generation. http://genealogysstar.blogspot.com/2014/12/what-percentage-of-worlds-records-have.html.

[7] Familytreedna. https://www.familytreedna.com/.

[8] Ged-inline. http://ged-inline.elasticbeanstalk.com/validate.

[9] Genealogy. http://www.genealogy.com/.

[10] The genealogy industry: $2 billionand growing! https://lisalouisecooke.com/2014/12/genealogy-industry-growing/.

[11] New england ancestors. https://www.americanancestors.org/index.aspx.

[12] Usgenweb. http://www.usgenweb.com/.

[13] Arne-cl. Bart coreference python wrapper. https://github.com/arne-cl/bart-coreference-python-wrapper, 2014. [Online; accessed 21-Oct-2017].

[14] R. Bishop. In the grand scheme of things: an exploration of the meaning of genealogical research. *The journal of popular culture*, 41(3):393–410, 2008.

[15] K. Clark and C. D. Manning. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667*, 2016.

[16] A. Culotta, A. McCallum, and J. Betz. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 296–303. Association for Computational Linguistics, 2006.

[17] Dasmith, Emilmont, and jcccf. Stanford corenlp python. `https://github.com/dasmith/stanford-corenlp-python`, 2014. [Online; accessed 21-Oct-2017].

[18] J. Efremova, A. M. García, J. Zhang, and T. Calders. Towards population reconstruction: extraction of family relationships from historical documents. In *First International Workshop on Population Informatics for Big Data (21th ACM-SIGKDD PopInfo15)*, pages 1–9, 2015.

[19] D. Embley, D. Campbell, Y. Jiang, S. Liddle, D. Lonsdale, Y.-K. Ng, and R. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data & Knowledge Engineering*, 31(3):227–251, 1999.

[20] D. W. Embley, D. M. Campbell, R. D. Smith, and S. W. Liddle. Ontology-based extraction and structuring of information from data-rich unstructured documents. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 52–59. ACM, 1998.

[21] J. Hansen. The coming web of genealogical data. In *2014-05-12]. http://fht. byu. edu/prev_ workshops/workshop 12/papers/3.1% 20Josh% 20Hansen% 20-% 20FHT% 202012% 20Workshop% 20Paper% 20-% 20The% 20Coming% 20 Web% 20of% 20Genealog-ical% 20Data. pdf.*

[22] D. Jurafsky. Text classification and naïve bayes. https://web.stanford.edu/class/cs124/lec/naivebayes.pdf, 2017. [Online; accessed 21-Oct-2017].

[23] P. Lindes, D. W. Lonsdale, and D. W. Embley. Ontology-based information extraction with a cognitive agent. In *AAAI*, pages 558–564, 2015.

[24] MACELab. Understanding genetic ancestry testing. https://www.ucl.ac.uk/mace-lab/debunking/understanding-testing, 2015. [Online; accessed 4-Dec-2017].

[25] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[26] C. Orasan, D. Cristea, R. Mitkov, and A. H. Branco. Anaphora resolution exercise: an overview. In *LREC*, 2008.

[27] L. Pasarin Perea. Analysis of alternatives to store genealogical trees using graph databases. Master's thesis, Universitat Politècnica de Catalunya, 2013.

[28] M. Poesio, S. Ponzetto, Y. Versley, V. Eidelman, A. Jern, A. Moschitti, X. Yang, K. Rodriguez, and O. Uryupina. Bart coreference and anaphora. http://www.bart-coref.org/, 2007. [Online; accessed 21-Oct-2017].

[29] Smartschat. Cort. https://github.com/smartschat/cort, 2017. [Online; accessed 21-Oct-2017].

[30] G. Team et al. The gedcom standard release 5.5. *Family and Church History Department, The Church of Jesus Christ of Latter-day Saints*, 1996.

[31] Thomwolf, Julien-C, and MaxwellRebo. Neural coref. `https://github.com/huggingface/neuralcoref`, 2017. [Online; accessed 20-Oct-2017].

[32] T. Walker and D. Embley. Automating the extraction of genealogical information from the web. In *Proceedings of Fourth Anual Workshop on Technology for Family History and Genealogical Research, Provo, UT*, 2004.

[33] F. Weil. *Family Trees: A History of Genealogy in America*, chapter 1. Cambridge, Mass.: Harvard University Press, 2013.

[34] C. J. Woodbury. Automatic extraction from and reasoning about genealogical records: A prototype. 2010.

[35] S. Woodfield. Effective sharing of family history information. In *Proceedings of the 12th Annual Family History Technology Workshop*, 2012.

APPENDIX: GEDCOM SAMPLE

```
0 HEAD
1 SOUR Wikipedia
1 GEDC
2 VERS 5.5
2 FORM LINEAGE-LINKED
1 SUBM @0@
1 CHAR ASCII
0 @0@ SUBM
1 NAME Timothy-Chu

0 @I1@ INDI
1 NAME philippine of /brandenburg-schwedt/
1 SEX M
1 FAMS @F0@
0 @I4@ INDI
1 NAME margrave frederick william of /brandenburg-schwedt/
1 SEX F
0 @I0@ INDI
1 NAME sophia magdalena of /denmark/
1 SEX M
0 @I8@ INDI
1 NAME friederike dorothea , duchess of /wrttemberg/
1 SEX F
0 @I2@ INDI
1 NAME /william/
```

1 SEX M

0 @I7@ INDI

1 NAME elisabeth louise , princess of /prussia/

1 SEX F

0 @I5@ INDI

1 NAME king frederick william /ii/

1 SEX M

1 FAMC @F0@

0 @I3@ INDI

1 NAME /charles/

1 SEX M

0 @I9@ INDI

1 NAME /wintzingerode/

1 SEX F

0 @I6@ INDI

1 NAME princess sophia dorothea of /prussia/

1 SEX F

0 @F0@ FAM

1 HUSB @I1@

1 CHIL @I5@

1 NCHI 1

0 TRLR