IMPROVING RELATION EXTRACTION FROM UNSTRUCTURED GENEALOGICAL TEXTS USING FINE-TUNED TRANSFORMERS

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

Carloangello Parrolivelli

June 2022

COMMITTEE MEMBERSHIP

TITLE:	Improving Relation Extraction From Unstructured Genealogical Texts Using Fine-Tuned Transformers
AUTHOR:	Carloangello Parrolivelli
DATE SUBMITTED:	June 2022
COMMITTEE CHAIR:	Lubomir Stanchev, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Foaad Khosmood, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Paul Anderson, Ph.D. Professor of Computer Science

ABSTRACT

Improving Relation Extraction From Unstructured Genealogical Texts Using Fine-Tuned Transformers

Carloangello Parrolivelli

Though exploring one's family lineage through genealogical family trees can be insightful to developing one's identity, this knowledge is typically held behind closed doors by private companies or require expensive technologies, such as DNA testing, to uncover. With the ever-booming explosion of data on the world wide web, many unstructured text documents, both old and new, are being discovered, written, and processed which contain rich genealogical information. With access to this immense amount of data, however, entails a costly process whereby people, typically volunteers, have to read large amounts of text to find relationships between people. This delays having genealogical information be open and accessible to all.

This thesis explores state-of-the-art methods for relation extraction across the genealogical and biomedical domains and bridges new and old research by proposing an updated three-tier system for parsing unstructured documents. This system makes use of recently developed and massively pretrained transformers and fine-tuning techniques to take advantage of these deep neural models' inherent understanding of English syntax and semantics for classification.

With only a fraction of labeled data typically needed to train large models, fine-tuning a LUKE relation classification model with minimal added features can identify genealogical relationships with macro precision, recall, and F1 scores of 0.880, 0.867, and 0.871, respectively, in data sets with scarce (~10%) positive relations. Furthermore, with the advent of a modern coreference resolution system utilizing SpanBERT embeddings and a modern named entity parser, our end-to-end pipeline can extract and correctly classify relationships within unstructured documents with macro precision, recall, and F1 scores of 0.794, 0.616, and 0.676, respectively. This thesis also evaluates individual components of the system and discusses future improvements to be made.

ACKNOWLEDGMENTS

- My advisor, Lubomir Stanchev, for being incredibly supportive, understanding, and extremeley knowledgeable for the various data science and machine learning aspects of this thesis.
- Committee member Foaad Khosmood for teaching me all about NLP + deep learning in his fantastic Speech and Language Processing courses at Cal Poly and offering valuable discussions and feedback for the models in my system.
- Committee member Paul Anderson whose teachings on machine learning and data science impacted me greatly through several courses.
- All committee members for being flexible and understanding in scheduling my defense.
- All Cal Poly CSC faculty I've taken courses with for enriching my knowledge immensely and giving me a truly one-of-a-kind education.
- My amazing parents Silvia Le Roux and Carlos Parrolivelli for supporting me, and specifically my mom Silvia for staying up late and listening to me pitch thesis ideas, rehearse my thesis defense, and offering to help in any capacity.
- My amazing friends Ghassan and Danny for commuting at 4 AM on my defense day from San Francisco just to see me defend and for staying up late with me on Discord while I worked on this thesis.
- My amazing friends Sam and Kira for giving me encouragement, support, ideas, and motivation whenever I needed it.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Genealogical documents are being collected, stored, and indexed at blazing speeds through commercial companies such as FamilySearch and Ancestry.com. While the general goal of such companies is to create large, accessible family tree data, having to parse through this data, digitize it, and understand it is very costly. This prevents ordinary citizens from reliable access to family records. Intelligent systems are being created to automate this indexing process [10], but very few accurate automated systems are being put to use to also extract familial relationship data from these documents. Thus, the problem exists that, while there exists an incredible amount of genealogically-rich documents, there is no way to quickly parse and understand them for creation of large family trees.

While structured documents such as census records, birth and marriage certificates, and others of the sort, follow very standardized formats that can be parsed by algorithms, unstructured documents with natural language describing relationships between people can generally only be understood by humans. Creating a system capable of parsing through and identifying relations between people, thus, would demonstrate that a system is capable of reasoning and understanding natural language on some level close to humans. Furthermore, a system to perform this would make genealogical history information infinitely more accessible to ordinary people, helping us to discover a stronger sense of our background and identity. The creation of such a system can also be extended to other areas of the world wide web such as blog posts, social networks, and online history databases to construct genealogical knowledge graphs from natural language input. These graphs could, in turn, make it easier for computers to reason about people and their relationships across the web.

As great as this sounds, though, recovering these relations from unstructured documents poses a very complex problem. In structured documents where there are clearly marked lines or sections denoting names, relations, dates, and other important attributes, unstructured text holds absolutely no guarantee of any of this. Because unstructured text is simply chunks of natural language, there are no rules to writing for authors outside of basic grammar rules. Furthermore, if they do contain the above attributes, they can be written and interpreted in a seemingly infinite number of different ways. Due to the nature of human language, rigid algorithms consisting of sets of grammar-parsing or syntax rules just cannot adapt to and interpret text the way humans can. These naive approaches are too limited in scope and do not develop any true understanding of the language or documents.

Prior approaches to solve the problem have attempted to create large numbers of rules and patterns for pattern-matching, but in texts that truly guarantee no structure, these methods always fall short and cannot guarantee future success. Furthermore, the lack of a large data set of golden-labeled data prevents robust training of deep learning models that must be trained from scratch. Works have combatted this by using distantly supervised data or feature-heavy machine learning approaches, though there is large margins for improvement throughout these algorithms. This thesis observes the landscape of relation extraction across genealogical and biomedical domains and turns towards massively pretrained transformer models that use self-supervised learning to develop deep understandings of syntax and semantics of the English text.

By adding a classifer layer on top of a pretrained transformer and fine-tuning the model, this thesis shows that having only a fraction of the data is enough to produce good relationship extraction results. Furthermore, this thesis details a standardized data collection process and evaluation process for components of a three-phase system to identify weaknesses for future work. Such three-phase systems for relation extraction of unstructured text have been used before, but the components are upgraded using as many transformer models as possible for best results. This, alongside a discussion of the components and their weaknesses, makes for a clear path of future work and encouraging results of a classifier with macro averaged precision, recall, and F1 scores of 0.880, 0.867, and 0.871, respectively, and end-to-end system macro averaged precision, recall, and F1 scores of 0.794, 0.616, and 0.676, respectively.

In all, this thesis makes four important contributions to the field of genealogical relation extraction. The first contribution is a hand-labeled data set with named entities, anaphors, coreferences, and relationships for 104 documents from Wikipedia. A discussion on how to label is included for future researchers to add to the data set. The second contribution is a novel algorithm for sanitizing the output for a transformer-based coreference resolution model. The third contribution is a fine-tuned, LUKE-based classifier that can predict genealogical relationships with a macro F1 score of 0.871. The fourth and final contribution is a set of evaluation methods and formulas for evaluating components and the entirety of the relation classification pipeline which future genealogical relation extraction works should follow.

Chapter 2 of this thesis details a comprehensive literature review of related works in genealogical, biomedical, and NLP domains for structured and unstructured texts. In Chapter 3, the problem is defined and the proposed system is introduced. Chapter 4 offers a discussion on the creation of an unbiased data set and Chapter 5 details how this data is used throughout all three components (and sub-components) of the system. Chapter 6 highlights the rigorous testing done on all components of the system and results, and Chapter 7 features conclusionary remarks and directions for future work.

Chapter 2

RELATED RESEARCH

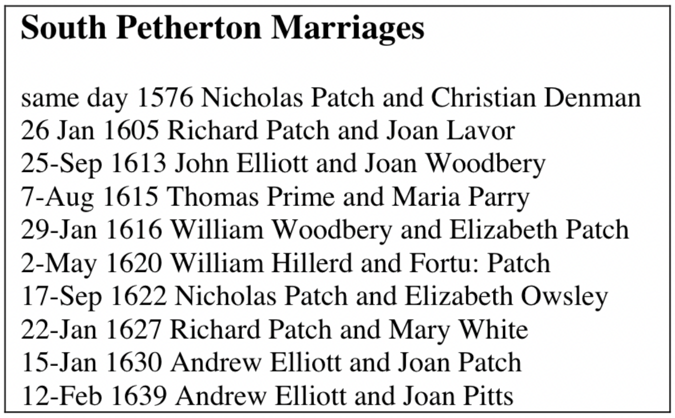
2.1 Genealogical Relation Extraction

Relation Extraction from natural language reaches a variety of different domains and is studied across many classes of data sets. Since this thesis takes inspiration from relation extraction across multiple domains, the discussion of related works will be separated by domain. This chapter will describe relation extraction as it has been applied to texts specifically containing genealogical family relationships between people entities.

2.1.1 Structured / Semi-Structured Text

There exists a great variety in guaranteed structure of historical genealogical texts, with a good amount making up documents classified as "structured" or "semi-structured" texts. These are typically official documents with rules regarding their formatting and processing.

An early work in the field of semi-structured text, [38] applies an ontology-based approach to genealogical structured text. Using documents from church records, parish records, and vital records across England, Denmark, and Massachusetts, respectively, the authors format them into HTML format from their origins (OCR, hand-transcribed, etc.) for use and extraction in the Ontology Extraction System (OntoES). An example of this data can be seen in Figure 2.1. The work uses lots of regular expressions from the domain-specific and expert-annotated ontologies to find


South Petherton Marriages

same day 1576 Nicholas Patch and Christian Denman
26 Jan 1605 Richard Patch and Joan Lavor
25-Sep 1613 John Elliott and Joan Woodbery
7-Aug 1615 Thomas Prime and Maria Parry
29-Jan 1616 William Woodbery and Elizabeth Patch
2-May 1620 William Hillerd and Fortu: Patch
17-Sep 1622 Nicholas Patch and Elizabeth Owsley
22-Jan 1627 Richard Patch and Mary White
15-Jan 1630 Andrew Elliott and Joan Patch
12-Feb 1639 Andrew Elliott and Joan Pitts

Figure 2.1: An example of a structured marriage document.

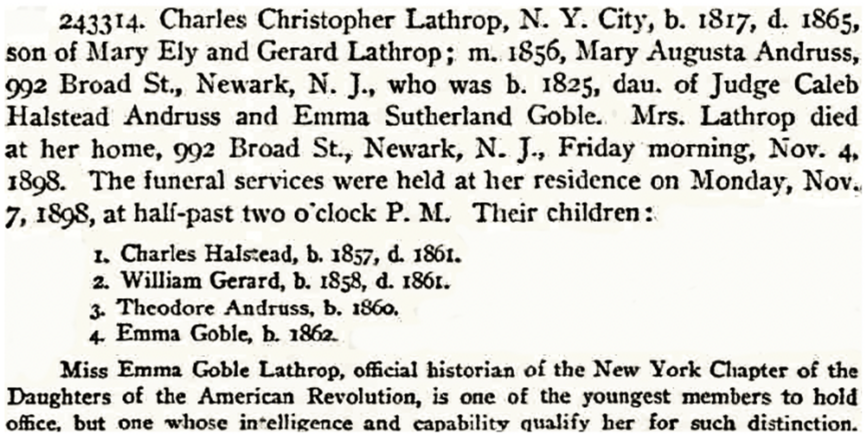
entities and relations in the documents. This reference shows that, for structured text, structured methods can work well.

A continuation of the ontology approach, [23] makes use of OntoES and a cognitive architecture named *Soar* as a means of "representing meaning and performing reasoning." The goal is that by using a combination of strictly rules-based parsing from the ontology and *Soar* to derive more semantic meaning, better results can be found with genealogical family relations that are more complex in nature. Though this work still relies on the structure of their corpus, which is family history books, a lot of the data they are parsing through moves away from strict structure to a "greatly abbreviated and highly formulaic style of English lexis and grammar." An example of this can be seen in Figure 2.2. Their results, which can be seen in Table 2.1, demonstrates that, as we move to more unstructured data, this problem becomes increasingly hard. This is nearing the scope of our thesis, which aims to work on completely unstructured text.

The work by [26] focuses on extracting family relations from literary texts by examining verbal interactions between characters and using their vocatives (how they address others), of which there are named vocatives (address by name), and nomi-

Author	Date	Unstructured	Structured	Corpora	# Relations	Rules-Based	Neural Model	P	R	F1
Genealogical										
[38]	2010		✓	Church, Parish, Vital records	N/A	✓		0.97+	0.88+	??
[26]	2014	✓	✓	Literary works (*Pride and Prejudice*	202	✓	✓	0.82	0.04	0.08
[23]	2015		✓	Family books	N/A	✓		0.93	0.68	0.78
[29]	2021		✓	Family books	N/A	✓		0.99+	0.98+	0.99+
[15]	1998	✓		Automobile ads, job listings	N/A	✓		0.98	0.84	0.90
[30]	2005	✓		Swiss documents	80,331		✓	N/A	N/A	N/A
[12]	2006	✓		Wikipedia	4,701		✓	0.71	0.54	0.61
[33]	2010	✓		Royal biographies, Portuguese newspapers	N/A	✓		0.71	0.24	0.36
[14]	2015	✓		Dutch historical notary acts	60,000+	✓	✓(SVM)	0.42	0.29	0.34
[14]	2015	✓		Dutch historical notary acts	60,000+	✓	✓(HMM)	0.82	0.59	0.69
[11]	2017	✓		Wikipedia	232,344		✓	0.02	0.90	0.04
[32]	2019	✓	✓	notarial acts, civil registers/certificates	N/A		✓	0.82	0.85	0.83
Biomedical										
[22]	2019	✓		THYME	13,285		✓	0.67	0.70	0.68
[40]	2021	✓		MADE1.0	27,175		✓	0.91	0.87	0.89
[40]	2021	✓		n2c2	59,068		✓	0.96	0.94	0.95
[18]	2021	✓		obituaries			✓	0.84	0.83	0.83
[17]	2021	✓		clinical notes			✓	-	-	0.88

Table 2.1: Summary and Scores of Related Works


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.

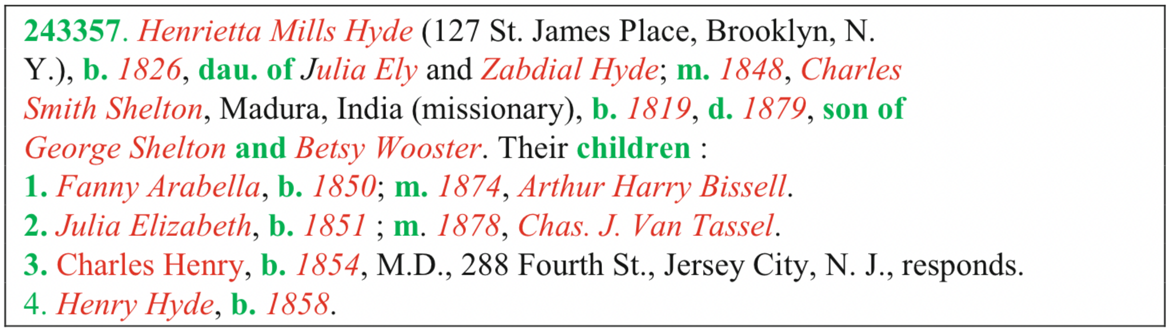
Miss Emma Goble Lathrop, official historian of the New York Chapter of the Daughters of the American Revolution, is one of the youngest members to hold office, but one whose intelligence and capability qualify her for such distinction.

Figure 2.2: An example of looser, structured text.

nals. They also use propagation to infer implicit family relations (two females with the same mother or father are considered siblings). In this thesis, this is considered semi-structured text because utterances do not always happen in all natural language and they are always marked by a set of quotation marks. Utterances are either attributed immediately based on context or features are made of the text and fed into a classifier (J48, JRip, Logistic Regression) with candidate speakers in close proximity. When utterances are detected, they only become candidates if they have a target nominal—a noun describing a family relation, such as "mother," "father," etc. The list of nominals are extended by using WordNet synonyms and hypernyms. A set of rules will create the relations and either an unsupervised or supervised approach will determine if an utterance is a vocative utterance or not (unsupervised pattern is <P my dear(est) T P> pattern where T is a target nominal and P is punctuation) and supervised uses a naive Bayes model with 10-fold cross validation. Furthermore, the authors write that they performed gender attribution and coreference resolution for named entities, but ground truth information was used for the present work. Results in Table 2.1 show decent precision and bad recall, demonstrating the difficulty of unstructured text. This thesis does not operate with utterances, though seeks to use a similar system involving entity recognition and coreference resolution. This thesis, however, lays no claim to gender assumption and parses normal natural language without dialogue between entities, mimicking the structure of historical documents.

A recent work by [29] develops an extremely robust, high-precision and high-recall system for historical family books, a form of structured text. Their data set comes from the 18th century Kilbarchan parish register [3], the early 20th century Miller funeral home records [5], and The Ely Ancestry [2]. In essence, their system is a two-part system that begins with labeling value phrases such as names and dates, and then links them together based on their sign phrases, such as dau. (daughter of), m. (married), etc. Also notable, each record begins with a "Head", and all other entities in the record may/may not be related to the head. An example of this is shown in Figure 2.3. After analyzing the structures of the books, the authors come up with four constraints that the structured text must follow regarding proximity of signs and values to each other and other records. With these constraints, word tokens are tagged based on alphanumeric format and noun and date configurations according to some templates in template-matching (regex). Relation Extraction can then be done, according to the authors, in a much simpler way because this tagging of labels to their entities automatically defines the relationships to the heads of the records. The authors present good results, but there are many distinctions from the work this thesis aims to do. Our system does not guarantee the presence of these "sign phrases" or any structure to link a "Head" to anyone else.

In all, this chapter demonstrates that structured text can be tackled with rules-based approaches using domain expertise, parsing rules, and regular expressions. The guarantee of structure allows for high-precision systems for automatic relationship extraction. Conversely, in this thesis we extract relations from unstructured text, where there are no constraints on positioning, grammar, or keywords in text.



243357. *Henrietta Mills Hyde* (127 St. James Place, Brooklyn, N. Y.), **b.** *1826*, **dau. of** *Julia Ely* and *Zabdial Hyde*; **m.** *1848*, *Charles Smith Shelton*, Madura, India (missionary), **b.** *1819*, **d.** *1879*, **son of** *George Shelton* **and** *Betsy Wooster*. Their **children** :

1. *Fanny Arabella*, **b.** *1850*; **m.** *1874*, *Arthur Harry Bissell*.

2. *Julia Elizabeth*, **b.** *1851* ; **m.** *1878*, *Chas. J. Van Tassel*.

3. Charles Henry, **b.** *1854*, M.D., 288 Fourth St., Jersey City, N. J., responds.

4. *Henry Hyde*, **b.** *1858*.

Figure 2.3: Color coded family book with entities and relation keywords highlighted

2.1.2 Unstructured Text

Paper [15] is an early work describing the use of a domain-specific ontology and a regular-expression based extraction method for identifying and relating objects in the ontology with their relations to other objects. Though this work uses ontologies and data from automobile advertisements from the *Salt Lake Tribune* and jobs listings from the *Los Angeles Times*, this work serves as a foundation that, given a detailed, domain-specific ontology, it's possible to get high metrics (precision > 0.98, recall > 0.84). The results were made based on "facts" from the documents, not explicitly relations.

Work [30], while not having the explicit goal of extracting Resource Description Framework (RDF) triplets of relationships between entities, attempts to tag each word in unstructured documents with a tag specified in Figure 2.4, one of which contains words that denote a relationship ("brother", "mother", etc.). They test both first and second order hidden Markov models (HMMs) that use an intelligent form of training that requires humans to manually annotate subsets of training data in increments until the mean error for predicting tags per word goes down. While there is a mismatch in the problem this work solves versus the problem this thesis tries to

Label	Description
First Name	First Name
Surname	A Surname
de / di	de, di, dou, du, etc.
comma	a comma
hyphen	The hyphen symbol '-'
Place	A place name
relation	mother, daughter, son, etc.
role	A profession or role within the community e.g. doctor, cantor
called	Used for aliases e.g. Anna *called Ave*
period	The period symbol '.'
(	Open brackets e.g. ({
)	Close brackets
and	the word 'et' and equivalent words
other	Any miscellaneous term

Figure 2.4: Tags used for labeling text using HMMs

solve, this work presents a foundation for using HMMs for relation extraction and proves their viability in tagging tokens in unstructured texts.

The work by [12] takes yet another hybrid approach, opting for a two-part system that uses a conditional random field model (CRF), a type of Markov network, in conjunction with a dynamic knowledge base to simultaneously extract relations in conjunction. The purpose of doing so is that classifying relationships with a CRF and linking those relationships in a knowledge base would allow for "implicit" relationships to be discovered that are not explicitly stated. The example they give for this is seen in Figure 2.5, where three explicit relations form one implicit relation amongst the involved entities. The construction of the knowledge base is a closed-loop system that involves building the base with the CRF and then using the base knowledge alongside each relation's confidence scores to extract implicit relations and prune incorrect relations. This thesis differs in the fact that no external knowledge base is needed with our approach. Our pipeline and model can find relations in data using context alone.

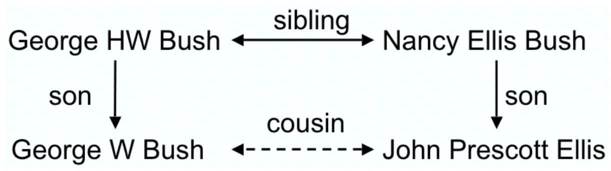


Figure 2.5: An implicit relationship shown in the Bush family.

The work by [33] contributed to the field of genealogical relation extraction in an important way. The aim of the authors was to improve familial relation extraction in Portuguese texts rather than English because there was very little work done on Portuguese texts at the time. They took a rules-based approach since there was no widely available Portuguese corpus big enough for use in machine learning models, extending on Named Entity Recognition (NER) models from other authors and creating 99 rules in Xerox Incremental Parsing (XIP) for extracting familial relationships. The rules revolve around identifying proper nouns, subject, predicates, and key words that could denote a relationship (father, uncle, etc.). Grammar rules were created on the training set to avoid bias. The author also notes that, while coreference resolution was considered for the pipeline, other researchers are working on it and that the problem of coreference resolution is "an unavoidable track parallel to relation extraction task." This work is one of the first to work on Portuguese texts, but more importantly showed that 1) with a lack of labeled data, using neural models becomes very difficult, which our thesis aims to disprove using pretrained models. Furthermore, this work shows that only using rules-based approaches (regular expressions, pattern matching) does not work well with unstructured text.

Paper [14] was a big advancement in the field of genealogical relation extraction from free-form texts. They use a Dutch data set made up of historical notary acts with types of documents such as "property transfer," "sale," "inheritance," "public sale," "obligation," "declaration," "partition of inheritance," "resolution," "inventory," and "evaluation." The NER system was custom and made use of a Dutch database of names as well as name labels (FN (first name), LN (last name), I (initial), etc.) that works in three phases. The first phase uses the name dictionary + tags and tags every word. The second phase uses regular expressions to include all patterns of possible names as official names. The third phase "disambiguates multiple occurrences of the same name into one," though the author notes it is uncommon to have multiple references to the same name, a big difference from our documents where they are likely many references to the same names.

As far as genealogical relation extraction, their first approach constructs feature vectors between every pair of names extracted from a document. These feature vectors include words in between and two words before and after first and last entities respectively. They calculate term frequency scores for each token, specifically not opting for Term Frequency-Index Document Frequency (TF-IDF) because they want to reward frequently occurring words. Since the vocabulary is large, they use bi-grams of words as a feature set. For this approach, multi-class and binary Support Vector Machine (SVM) classifiers are made and trained.

The second approach uses a hidden Markov model to tag each token in a string with tags that denote either words in a name phrase, relation phrase, or other words using a system with three prefixes, B, I, and O. Name and relation descriptors will start with B-(PER/REL), followed by I-(PER/REL) for each subsequent relation token, ending with O and filling O for all other miscellaneous tokens. Then they can link together relations using regular expressions / grammars to extract relations, such as [PER, REL, PER] or [PER] + 'en'[PER]',’[REL] [14]. They perform two experiments, where each uses the relationship tags but one uses the person tags as well and the other instead only labels relation tags and uses the NER from the first model's pipeline, replacing names with a single PER tag. The authors use data augmentation that inserts relation phrases ("married to", "children of", etc.) alongside auxiliary tokens according to pattern grammars. This created an extra 10,000 documents and 907,000 words.

This work marks a shift in relation extraction in a genealogical sense. Introduced here are the use of two models, the SVM and the HMM, as components of classification pipelines. This thesis differs in that the amount of data used is lower, with deeper models, and very importantly uses a coreference resolution system as opposed to this work that just uses an NER system. Furthermore, this thesis shows that data augmentation need not be utilized by using a fine-tuned approach, and that the emphasis is on quality of data, not quantity.

One of the most recent advancements in the field of genealogical relation extraction, [11] creates a data set made up of 339,248 documents containing 232,344 relationships from Wikipedia. This data set is created in a "distantly supervised" manner, using WikiData alongside proximity and occurrence of related entities to predict the presence of relationships within text. Effectively, this creates training data that is potentially correct. From this, they train a multi-class Naive Bayes classifier and binary Naive Bayes classifiers for each class, using balanced and unbalanced subsets of the data set + named entity recognition + a naive anaphora resolution system to create feature vectors for the model for training. This use of data generation makes for the use of deep neural models possible, however their results on classifiers for unbalanced data is very low due to low precision rates. This thesis aims to take an opposite approach of using small, extremely accurate "gold-labeled" (hand-labeled, correct) data to fine-tune models and purposefully optimize for heavily imbalanced data sets.

Another work [32] takes a similar hybrid-training approach in which they use a "distant supervision" approach to generate highly precise (0.90 precision) training data for use in training models to classify relations in never-before-seen unstructured texts.

They do this by creating tuples of combinations of entities found in the unstructured texts, creating "fingerprints" for the tuples and their possible relationship, using a Levenshtein distance to find neighbors in the knowledge base, and a statistical assessor to determine whether the entities are actually related or not according to the relationships of their neighbors. Once relationships are deemed probable by this system, the authors can simply extract the features for the relationship (text before, in-between, after, and total length of words) from the unstructured text and add it to a training set. Using this newly formed distantly supervised training set, they train a decision tree classifier on two testing sets, of which the first has time overlaps (1800-1912) with documents in the knowledge base and the second has no time overlap (1700-1800) with documents in the knowledge base so that there is no bias leakage. This approach shows a more efficient way of creating more accurate unsupervised data, but this thesis still differs and hypothesizes that using data that is improperly annotated can still very much negatively impact the overall system.

Overall, in the genealogical domain, this timeline of related works shows that, while there are trends to move towards statistical models over rules-based approaches for unstructured texts, there is always a lack of enough high quality data for training. This leads to approaches attempting to create data through the use of external knowledge bases. This thesis differs from these works by, instead, choosing new state-of-the-art pretrained models that inherently have learned an understanding of language. In doing so, this thesis can use less data to train and achieve higher accuracy with less features.

2.2 Biomedical Relation Extraction

The field of relation extraction from genealogical free-form texts is slow-growing and so this thesis takes as inspiration works from the biomedical domain where state-of-the-art relation extraction across several domains of entities and relations are taking place.

A paper by [22] Was one of the earlier works that applied Bidirectional Encoder Representations from Transformers (BERT) [13] to relation extraction in the biomedical domain. The relation extraction task involves identifying CONTAINS relations of EVENTs within a narrative container. The authors tested several BERT configurations, including BERT models pretrained on biomedical corpora like BioBERT and BERT-MIMIC alongside the normal BERT pretrained model and added entity annotations for them. Their research shows that these BERT models, when non-XML entity annotations are used (WordPiece will break down XML entity annotations into multiple tokens and custom vocabulary tokens need pretraining to contextualize their representations), the inherent language understanding and hidden embeddings learned by BERT can result in systems that outperform prior state-of-the-art systems that use bidirectional long short-term memory (bi-LSTM) [19] models. The authors attribute this to the fact that BERT models 1) can handle cross-sentence examples instead of being limited to relation extraction (RE) within sentences because BERT does not need sentence boundaries to operate 2) the bidirectional attention model on BERT plus its pretraining allow tokens to have embeddings that are not immediately dependent on their closest neighbor tokens, and 3) since bi-LSTMs are not pretrained they suffer from a limited example of gold annotations.

Paper [40] moves forward in prior work for RE in a clinical setting and tests BERT, RoBERTa, and XLNet (all variations of the same transformer architecture) and their medically pretrained counterparts (BERT-clinical) on two publicly available clinical RE data sets from two challenges, namely the 2018 MADE1.0 challenge and 2018 n2c2 challenge. The first challenge exposes a roughly 80-20 split of 1090 notes forming 27175 relations, while the second challenge has a roughly 60-40 split of 505 notes with 59068 clinical relations. These data sets are made from 21 cancer patients' clinical notes and the relations include features, such as relations for medication and medication attributes (dosage), medication signs and symptoms, and severity. The latter challenge has six categories for drugs and drug attributes and eight relations overall. Relations can be found spanning one or multiple sentences. All clinical concepts are assumed to have been identified already and the authors use a heuristic-based concept pair generation method where only certain concepts within reasonable types can be related (for example, a drug concept and dose concept can form a pair but an Adverse Drug Event (ADE) concept and dose concept cannot form a pair).

For classifying models, one approach was taken as a multiclass classifier where all relation categories were added alongside a "non-relation" category and another approach was a binary classifier for the existence of any relation and then a rules-based parser to determine the relation based on the types of clinical concepts. For cross-sentence relations, one approach taken was training a classifier to handle all relations regardless of length and another approach was a distance-specific that classifies relations within candidate pairs that span the same number of cross-sentence boundaries (0 = same sentence, 1 = relation across 2 consecutive sentences, etc.). Softmax layers were applied to the ends of the models. Entity markers `[S1]`, `[E1]`, `[S2]`, and `[E2]` were inputted and the contextual representations of the `[CLS]` token and all four entity markers were fed into the final classification layer. Transformers learn representations at token-level and sentence-level (`[CLS]`). The multiclass classifiers, those most relevant to this research since the relations cannot be deduced by entity type in genealogical texts, have very high scores with RoBERTa-clinical scoring the best overall. This study proves that more robust models, as well as using other token representations such as entity markers, can prove very efficient results for RE in the clinical domain.

Paper [18] is also a very important work in the biomedical domain, as it seeks to improve electronic health records (EHRs) with patient-relevant genealogical family tree data using unstructured text sources. They develop an end-to-end automatic extraction system that combines named entity recognition (NER) and relation classification (RC) into a single step (amongst also matching locations to people in Step 2, expanding last name distributive rules in Step 3, recognizing spouse names in parentheses in Step 4, and inferring age, death date, or birth date using rules in Step 5). It's important to note that the data set that the authors use is made of 12,047 obituaries from funeral service websites and a local newspaper in Minneapolis. 1,700 were randomly sampled to be used in the corpus by three annotators, consisting of 71 types of relationships and four types of entities (names, residences, birth dates, and death dates). Since obituaries are of a certain person, within an obituary's metadata you can find the "baseline entity," of which the authors made all other entities in the passage related to. Thus, this work does not capture relations between entities in text to each other, only the baseline entity, unlike the goal of our thesis. Step 1 is of most importance to this thesis as this thesis does not aim to tackle locations or dates, strictly just relations between entities. Paper [18] used a tagging scheme where entities could be labeled by their start and end tokens as well as their relationship (for example, someone's sister could be labeled ["sister_B", "sister_B", "sister_E"] where B, I, and E indicate the beginning, inside, and end of a named entity that is also related as the sister to the main entity. Importantly, their work uses BERT to obtain encoder representations for each token they label. Learning the correct representations for the NER + RC tokens vs. other tokens are part of the reason their system is so successful as compared to something such as TF-IDF. To combat the issue of data imbalance from more frequent relations like "child," which had 2489 instances in their corpus, compared to something such as "great-grandson," which only had 58, the authors used data augmentation to generate more training examples using key "relationship tokens" and randomly synonym replacing, inserting, swapping, or deleting tokens. In evaluating their results, they calculated macro and micro averaged P, R, and F1 in 10-fold cross validation. Their work is likely the most similar to ours in the use of k-fold cross validation, macro averaged scores, a multiclass classifier, and the NER + classifier pipeline. What differentiates this thesis and [18] is the use of a coreference system to capture **all** all relations within a text between any entities and the use of BERT-based models that can reason about entities rather than just using entity markers.

Lastly, another work by Harnoune et al. [17] demonstrates the great performance of an end-to-end system using BERT-based models in several parts for a general form of relation extraction in the clinical domain. The pipeline starts with preprocessing and HuggingFace's NeuralCoref for coreference resolution, then BERT contextualized embeddings are obtained for all of the tokens (WordPiece tokenizer) and a BERT-based model (BioBERT, BioClinicalBERT, BioRoBERTa, etc.) is used with a conditional random field (CRF) layer on top for named entity recognition. These entities are then fed back into a BERT-based model after an entity-specific masking is done and the `[CLS]` token's representation is fed into a fully connected neural network for binary classification (softmax) to determine if there is a relation or not. The data set is made of 505 openly accessible patient clinical notes, and the nodes (entities) supported by this system are Posology (prescription of drug and info), Drug, Patient, Reason, and ADE. Though their work's relations can be reduced to binary classification because the interactions between entities are specific, their work shows the potential use of an NLP pipeline consisting of NER, coreference resolution, and BERT-based models for contextual embeddings for robust relation extraction. This thesis differs in that the classification we perform is more sophisticated because relations between entities cannot be deduced solely from the type of entity.

A comprehensive summary of all of the beforementioned works and their results, corpora, approaches, etc. can be found in Table 2.1. The works in the biomedical domain go to show that, although the relation extraction tasks differ significantly in both biomedical and genealogical domains, works in the biomedical field are successfully using NER and pretrained transformers to achieve soaring high results in this task. This thesis uses this trend to find and use pretrained transformers in the most efficient method for genealogical relation extraction.

2.3 BERT, RoBERTa, SpanBERT, and LUKE

Contextualized word representations and its research has taken off in the latest few years among the NLP community. Previously state-of-the-art models like Embeddings from Language Models (ELMo) [31] found that by pretraining deep language models consisting of deep neural networks, such as forward and backward long short-term memory networks, deep models could "learn" representations for words, producing similar vectors for semantically similar words. The sequential nature of these networks, however, made pretraining more expensive and impossible to parallelize, hindering these networks ability to process longer training samples. [35] changed this narrative by removing the sequential model, training with positional encodings for words, and using stacked self-attention mechanisms that allow words to "see" other words in sentences for context-specific training. This creation of the transformer model allowed for massively paralleled training and increased learning of long-range dependencies.

Paper [13] used transformers and their parallelizability to create Bidirectional Encoder Representations from Transformers (BERT). In essence, rather than training transformers on specific tasks, [13] sought to develop a model focused on language understanding by using self-supervised training on massive corpora to, like ELMo, learn deep representations for words based on context. The research was made possible due to lower computational costs and parallelizability of transformers. The authors proved that once a large model develops deep understanding of language semantics and syntax, an additional output layer can be added to the end of the model utilizing fine-tuning for what the authors refer to as "downstream tasks," which can include any NLP problem, to attain state-of-the-art performance. This thesis takes the approach of fine-tuning a BERT-based model and confirms this.

BERT used two pretraining tasks, listed below:

1. Masked Language Modeling (MLM) - 15% of WordPiece tokens in sequences are masked at random with either a `[MASK]` token (80%), random token (10%), or an unchanged token (10%). The model must learn to predict the missing words. This allows for truly bidirectional learning since the word cannot be "seen" by itself from either direction, and this allows the model to learn syntax and semantics with no human intervention.

2. Next Sentence Prediction (NSP) - Pairs of sentences A and B are chosen where B is the true next sentence to A 50% of the time. The model must learn to predict when sentences are correctly related and should be consecutive. This task helps with future tasks such as question answering (QA) and natural language understanding (NLU) that require understanding relationships between whole sentences rather than just words. [13].

Paper [24] explored the pretraining tasks further, finding that BERT was under-optimized and released "A Robustly Optimized BERT Pretraining Approach" (RoBERTa). Their study developed RoBERTa by pretraining transformers in a similar fashion to BERT, but improving the MLM procedure to dynamically mask tokens during pre-training, rather than masking tokens in the training set before pretraining, removing the NSP learning objective, and training BERT for longer on bigger batches, among other techniques. These optimizations achieved comparable or better results. Paper [20] looked to optimize BERT as well, though specifically for NLP tasks where reasoning between spans of text, rather than just words, was more important. The authors changed BERT's pretraining approach to mask contiguous spans of text rather than WordPiece tokens and added a new span boundary objective (SBO) involving predicting tokens in masked spans using tokens at the boundaries of the spans. This new model was called SpanBERT. For tasks such as coreference resolution, this has produced groundbreaking results.

Though these methods work very well with fine-tuning an out-of-the-box model for tasks such as text classification or question answering, relation extraction poses a unique issue that feature-heavy models do not deal with. Since BERT-based models include special tokens during pretraining (`[CLS]` at the beginning and `[SEP]` in between sentences or statements), predictions for text classification and question answering can make use of the representations for those special tokens to derive knowledge, but relation extraction involves reasoning about pairs of entities and their contexts to derive relation knowledge. Furthermore, in sentences with many entities, there is no standard token or representation differing entities from other tokens, meaning that using the output of the `[CLS]` token can give us arbitray and unrelated relation representations. Though there has been research in using special entity marker tokens, namely in works such as [9], and research in including entity position embeddings [41], there is variation in what markers to use and how BERT-based models use these markers. Furthermore, since BERT-based models use WordPiece tokenization, if entity markers are not included in pretraining, then they must be added to the vocabulary manually, giving them random vector representations with little to no meaning, or they will be broken down into several WordPiece tokens. Pretraining BERT models is an expensive task that defeats the purpose of fine-tuning pretrained models for specific tasks. With feature-heavy approaches, locations of entities and information about them in relation to other entities can be easily engineered.

A new BERT-based model, Language Understanding with Knowledge-based Embeddings (LUKE) [39], solves the issue by changing, yet again, the pretraining tasks that BERT uses. The model focuses on building representations for both words and entities by introducing a new pretraining task that aims to predict randomly masked words *and entities* as well as introducing an entity-aware self-attention mechanism to consider types of tokens in computing representation scores. Furthermore, the work extends training on a RoBERTa model, incorporating state-of-the-art optimizations alongside entity representations. The new entity-aware self-attention mechanism allows this thesis to compute representations for entities in genealogical text and properly reason about their relations without having to worry about pretraining new BERT models with special entity markers or special embeddings. Furthermore, when computing entity representations for classification tasks, HuggingFace supplies a LUKE model where there is a linear layer placed on top of the hidden representations of both entity tokens. The entities are masked using two special mask tokens, namely `[MASK1]`, `[MASK2]` (the original paper uses `[HEAD]`, `[TAIL]`, but the source code has since been updated), which prevents overfitting by learning relations based on the names of the entities.

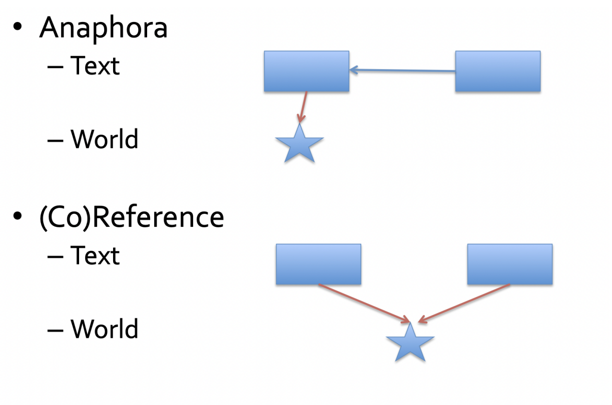


Figure 2.6: A visual difference between anaphora and coreference resolution

2.4 Coreference / Anaphora Resolution

To get some confusing terminology out of the way, coreference resolution and anaphora resolution are related but differ slightly. In coreference resolution, the task is to resolve mentions that refer to the same real-world entity, while in anaphora resolution the task is to resolve mentions to their antecedents and the interpretation of the mention is determined by the interpretation of the antecedent [27]. Thus, anaphora resolution is typically seen as pronoun resolution, but anaphors can present themselves in 3 ways:

1. Pronominal - referent is referred by a pronoun,
2. Definite noun phrase - antecedent is referred by a noun phrase, and
3. Quantifier/Ordinal - anaphor is a quantifier like “one” or an ordinal such as “first.”

[34]

A diagram of this difference can be found in Figure 2.6 [27].

Barack Obama nominated Hillary Rodham Clinton as his secretary of state on Monday. He chose her because she had foreign affairs experience as a former First Lady.

Figure 2.7: Sample text with highlighted references to the same real world entities.

For the domain of genealogical texts, the problem usually boils down to resolving anaphors in the pronominal and definite noun phrase types. This thesis finds that the best performing coreference resolution models simultaneously perform coreference and anaphora resolutions, but for simplicity and consistency with academic papers, this thesis refers to the model used as a coreference resolution model. A sample paragraph with all entities and mentions that should be resolved to each other is in Figure 2.7

There are two prominent coreference resolution systems in use by many researchers. They are HuggingFace's NeuralCoref [37] [6] and AllenNLP's coreference resolution model [1]. Under the hood, both models take deep-learning approaches to the problem of coreference resolution. Both models are trained and evaluated on the OntoNotes 5.0 dataset [36], which contains roughly 1.5 million annotated English words, among around another 1.5 million more for other languages combined.

HuggingFace's NeuralCoref model is a deep-learning method utilizing custom word embeddings with the ability to calculate embeddings for unknown words "on-the-fly" [8] by averaging embeddings for related words (for example, "Kendall Jenner" can be calculated from averaging vectors for "woman" and "model"). Furthermore, to give mention spans some contextual information, word vectors around the mention are averaged and used as the span embedding. These "contextualized" embeddings and embeddings for tokens within and around the spans are fed into two neural networks alongside other features, such as length and location of mentions. The first neural network takes these features and gives a score of the mention referring to the antecedent, and the other neural network gives a score of the mention having no antecedent. Highest score wins. To top it off, the model is built on top of SpaCy, a very popular natural language processing tool, to make use of their fast tokenization and parsing.

While the model works well, research on embeddings for words and spans have advanced significantly, making way for new state-of-the-art models like AllenNLP's coreference resolution model. The first improvement the model has over NeuralCoref is the use of bidirectional long short-term memory (LSTM) networks alongside fully connected feed forward neural networks to compute vector representations for span, antecedent, and head representations. Originally, the model used GloVe embeddings, which are context-insensitive word embeddings, but as of now has been improved to use state-of-the-art SpanBERT embeddings. SpanBERT's optimizations for span representations ties in beautifully with coreference resolution as the entire goal is to compare vectorized representations of different spans of text that might refer to each other. Furthermore, embeddings in BERT-based models are truly bidirectional and context-aware of their input sentences, a huge improvement from prior context-insensitive embeddings (GloVe) or averaged context embeddings such as those used in NeuralCoref. lastly, the model uses a coarse-to-fine inference which allows aggressive pruning of antecedents with low scores and a much lighter computational load to compute more pairs of span scores over longer distances.

Given a comprehensive overview of related literature in genealogy and a comparison to works in the biomedical field, we can now identify the problem we want to solve and propose a solution. The solution this thesis takes is inspired by works in both domains and aims to catch genealogical research up to the state of the art. At a high level, the steps of the solution are not unique, but their use together with new transformer models present a novel pipeline to solve our problem.

Chapter 3

PROBLEM DESCRIPTION AND PROPOSED SOLUTION

3.1 Definition and Desired Output

In essence, the problem this thesis aims to solve is that of building a system that can parse unstructured text in multi-paragraph form consisting of English natural language to build family trees of all human entities within. An example input and output of this problem can be seen in Figure 3.1. While there is other research being done in other languages [33], the domain of this problem is restricted to English. The end family trees link together entities with their genealogical relations to one another. It is important to distinguish that in this problem definition, the system does not aim to build relations only to a principal entity and their relationships, unlike similar research. The goal is to find relations for any and all pairs of entities that have them so long as they are genealogically related (many other works define relations pertaining to living in locations, drug-reactions, etc.). In the thesis, the problem will only require labeling of four major relationships categories, namely "parent-of," "child-of," "spouse-of," and "sibling-of," as these are typically seen as the most important / relevant for genealogical and biomedical applications. The problem also, unlike other related works, requires that the relationships be unidirectional when possible, meaning the "parent-of" and "child-of" relations must be unidirectional because the direction of the relationships defines the relation of one entity to another. Furthermore, relations should only be labeled if they can be explicitly deduced from the text, meaning that if two entities had a child, for example, there may or may not be a guarantee that those two entities were married to each other ("spouse-of").

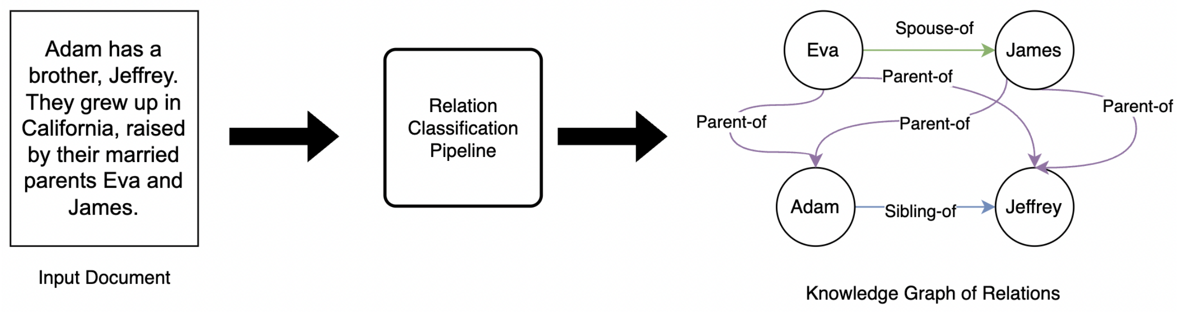

Adam has a brother, Jeffrey. They grew up in California, raised by their married parents Eva and James.
Input Document
Relation Classification Pipeline
Eva
James
Adam
Jeffrey
Spouse-of
Parent-of
Parent-of
Parent-of
Parent-of
Sibling-of
Knowledge Graph of Relations

Figure 3.1: A high-level overview of the problem and its solution

Avram Noam Chomsky was born on December 7, 1928, in the East Oak Lane neighborhood of Philadelphia, Pennsylvania. His parents, Ze'ev "William" Chomsky and Elsie Simonofsky, were Jewish immigrants. William had fled the Russian Empire in 1913 to escape conscription and worked in Baltimore sweatshops and Hebrew elementary schools before attending university. ...

Noam was the Chomskys' first child. His younger brother, David Eli Chomsky, was born five years later, and worked as a cardiologist in Philadelphia. ...

Figure 3.2: A paragraph of unstructured genealogical text containing relations between several entities.

While Figure 3.1 shows this process graphically, Figure 3.2 shows a specific example entry for Avram Noam Chomsky obtained from Wikipedia. From the text, we can see that there are several entities including Avram Noam Chomsky, his parents, and a brother. The end goal would be to produce a knowledge graph (KG) of all the related entities with directional edges that denote the type of relationship. An example of this is shown in Figure 3.3. Note that in this example, Chmosky's parents do not have a relationship to each other because, though it is assumed they were married to have Chomsky and his brother, it is not explicitly stated. The goal in solving this problem is to only find relationships that are explicit, with no assumptions made, in order to maintain the most historical accuracy.

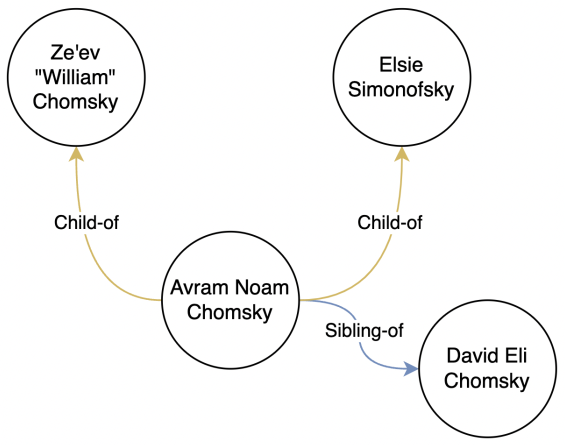


Figure 3.3: The genealogical graph of Chomsky's sample text

(Avram Noam Chomsky, **child-of**, Ze'ev "William" Chomsky)
(Avram Noam Chomsky, **child-of**, Elsie Simonofsky)
(Avram Noam Chomsky, **sibling-of**, David Eli Chomsky)

Figure 3.4: Several example RDF triplets.

With this data, resource description framework (RDF) triplets can be made from the nodes and edges on the graph. Figure 3.4 shows the possible triplets that can be extracted from the end graph.

3.2 Motivation

Though there are many structured documents describing historical genealogy between people in history, with the expansion of the world wide web (WWW), access to blog posts, online articles, and newly digitized historical documents using optical character recognition (OCR), there are a seemingly infinite amount of new texts that follow no patterns, but contain rich genealogical information. Humans can solve this problem with extremely high precision, but it takes a lot of time to read and understand texts and relate people with each other. This laborious process is typically done with hundreds of volunteers, but even then the growth in genealogical history is slow.

Instead of taking a human, say, a few minutes to read a document and relate entities, an automated system could perform the same task in mere seconds and it could do so "around-the-clock." This increase in efficiency could make access to historical records and genealogical family trees more accessible for people to discover their family history.

Furthermore, some genealogical information is held behind private doors of companies like 23AndMe that require expensive DNA testing kits and rely on a centralized database privately made methods for testing and relating DNA. Data privacy concerns might also arise due to the handling of such sensitive data. Automating this system for unstructured text, given that we already have great rules-based parsers for structured text, would help remove these restrictions for people to view their genealogies.

Finally, as we have seen from the biomedical papers, the ability to create genealogical histories for patients from clinical notes [17] or obituaries [18] could vastly improve the quality of healthcare that can be provided. Having access to blood-related family without a doctor having to read hundreds of pages of genealogical texts for their patients could serve to lessen pressure on the healthcare system and provide better, more accessible coverage for all.

3.3 Proposed Solution

The solution this thesis provides aims to supply the most general-purpose system capable of extracting relations from any and all entities mentioned in chunks of unstructured text. The main components of the system is a pipeline of 3 steps, shown in Figure 3.5, that work in unison to tag full entity names, shortened entity names, and their pronouns and mentions, and provide relation classifications both within sentences and across sentences.

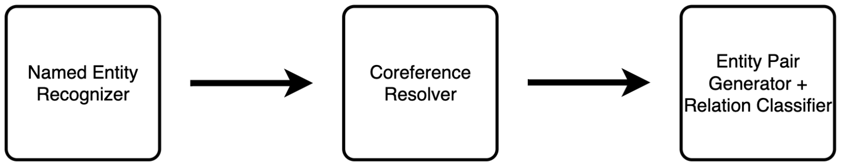


Figure 3.5: The main three components of the proposed pipeline

The first part of the pipeline makes use of a Named Entity Recognition system that attempts to parse out primarily full names within a text and the most amount of shortened names possible. The second part of the pipeline is a coreference resolution system that identifies named entities and their most probable pronouns and mentions. The resulting entities from these parts of the pipeline are cross-referenced with each other and the coreference mentions are collapsed into flattened trees with their primary entity. Given this information about the text, the third part of the pipeline attempts to create the shortest distance, most contextually relevant relation sentences that it can and classify the type of relation within the relationships "parent-of," "child-of," "spouse-of," "sibling-of," or "no-rel" for no relationship existing.

This approach attempts to validate two hypotheses that solve two problems in the field of relation extraction within the genealogical domain. The first is that, because proper nouns representing named entities can at times be sparse within a document (many mentions of a person might be replaced with pronouns, for example), the detection of these entity *mentions* and the ability to relate them can serve to create more contextually relevant relations than if names alone were used. This is in contrast to many related works that skip the coreference resolution step entirely. For example, in Figure 3.6, the text between the two people, Charlie Sheen in blue and Emilio in pink, is large and includes many other context clues and entities that might confuse a classifier. The hypothesis is that if we can find the closest entity mention to Emilio from Sheen, it could provide text that more clearly demonstrates a relation. In this case, the most representative statement is "his brother Emilio." This is because of a

Sheen attended Santa Monica High School in Santa Monica, California, along with Robert Downey Jr., where he was a star pitcher and shortstop for the baseball team. At Santa Monica High School, he showed an early interest in acting, making amateur Super 8 films with his brother Emilio and school friends Rob Lowe and Sean Penn under his birth name.

Figure 3.6: A sample paragraph with large distance between two relation-having entities, but short distances between entity 1's pronouns and the related entity.

sub-hypothesis this thesis makes that the vast majority of relations that can be found in text where at least one mention of an entity is in close proximity to the name of entity it is related to. While there are many instances were two proper names can be in close proximity and define a relationship, there are many instances where there is a closer identifier of a relation.

The second hypothesis is that by using a massively pretrained transformer model, such as BERT or LUKE, we can make more intelligent, context-aware classifications thanks to its inherent understanding of language syntax and semantics from its pretraining and contextualized word embeddings, and we can also attain robust results with only a fraction of the data typically needed to train deep neural networks. Deep neural networks are extremely data-hungry, often needing tens or hundreds of thousands of examples to attain reasonable results. In a field like genealogical relation extraction, where there is a limited amount of labeled data, this is an issue that has deterred research in deep neural networks (as seen by the lack of recent papers making use of them in this field). Because pretrained transformers, such as BERT and LUKE, are trained in a self-supervised manner on millions of training examples [13], their weights denote a natural understanding of the language. This means that by applying transfer learning and attaching a classification output layer to the end of a transformer model, these weights can be used to fine-tune the final layer on a much smaller subset of data to get powerful results.

This thesis theorizes, however, that to do this efficiently a set of gold-labeled data must be carefully made to steer the model in the right direction. The process of obtaining this data will be described in the next chapter. Special care to the pitfalls of including bias and creating bad quality data are discussed, as a model can only be as good as the data it trains on.

Chapter 4

DATA SET GENERATION

This thesis uses a corpus of 104 hand-labeled documents spanning 777 relations. The breakdown of the relations and their types can be seen in Table 4.1. The labeling software used for this thesis was Label Studio, one of the largest and most supported open-source labeling programs [4]. The data source chosen for these documents is Wikipedia because it is widely accessible, rich in genealogical information regarding people, and is a crowd-sourced information base. Since anyone can edit Wikipedia pages and many pages are written by multiple authors, by using data from many documents written in this fashion, this data set hopes to encompass a large breadth of writing styles, word choices, and sentence structures. In using this highly-diverse data set, the hope is that models trained and tested on the corpus can be applied and relevant to many other unstructured genealogical texts, regardless of the variation in writing. This is a polar opposite approach from many of the related works that aim to build rules or make assumptions on stylistic choices within texts to extract the most number of precise relations. Our solution tries to make a system that is widely applicable to many documents rather than a subset of documents only. The rest of this chapter will describe characteristics of the data set, how documents were chosen, and how data was labeled so that future researchers can pick up where this work left off and crowd-source a golden genealogical data set.

Label	# Relations
No-rel	6,458
Child-of	227
Parent-of	205
Sibling-of	178
Spouse-of	167

Table 4.1: Breakdown of labeled relations

4.1 Diversity and Inclusion

One of the biggest dangers in training supervised machine learning models is that the quality and characteristics of incoming training data will be reflected in outgoing predictions—for better or for worse. Researchers, then, must have extreme caution in creating non-discriminatory data that takes into account a diverse set of whatever population they are capturing data on, especially when dealing with human entities. This thesis makes it a point to create a non-bias, non-discriminatory corpus by taking into account the gender and ethnicity of the entities mentioned in documents. Since Wikipedia entries are usually written with a main or principal entity in focus, this corpus has roughly half of its documents belonging to male entities and half belonging to female entities. This thesis also aims to balance the relations of entities within the documents by gender as well, for example ensuring that there are many occurrences of the "sibling-of" relation that encapsulate both brothers and sisters, or "parent-of" relations that capture fathers and mothers.

While this thesis did attempt to incorporate more non-binary identifying entities, genealogical data was typically much more scarce. Furthermore, modern coreference resolution systems have difficulty relating gender-neutral pronouns, such as "they" and "them," for example, to their respective non-binary person entity. This work relies on a purely pretrained coreference resolution model, so these errors are assessed individually and in the end-to-end pipeline and discussed in the results.

This thesis does, however, make relations gender-neutral so as to not have some relations that are strictly gendered and some that are non-gendered when a gender cannot be assumed or used. In creating the corpus, there were several instances where the gender of related entities was not clear from the text, and so making relations gender-neutral combat this ambiguity by choosing not to assume the gender of an entity, in contrast to prior works that build anaphora resolution systems that try to make this assumption through classifiers. Though this makes genealogical data less fine-grained, assuming genders and splitting data across more labels is likelier to obfuscate a model trying to extract the contextual representations of relationships amongst small data sets, such as our corpus.

4.2 Selecting documents

Other important issues facing the creation of data sets for training supervised models are those of overfitting. The genealogical corpus aims to reduce the chances of overfitting by consciously choosing documents whose principal entities are minimally related to each other. The argument could be made that if all the documents in a corpus relate to the same family or a series of related families, then the system is learning to model those families and their relations rather than building a fundamental understanding of relations between any entities depending on context. Thus, in creating this corpus, this thesis takes the approach of performing depth-first searches (DFS) across mentioned entities within a document's text and restarting the search randomly throughout the labeling process. People entities that are unrelated are prioritized in the search to reduce familial overfitting. Walking through an example could be helpful.

Assume we start at a random entity, say Avram Noam Chomsky's Wikipedia page. We could find and label many relations to other entities found within that document. Of the other entities found, we could have a "child-of" relation to Elsie Simonofsky and "sibling-of" relation to David Eli Chomsky. Some other entities found in the document that *do not* have a genealogical relation could be Barbara Partee, a collaborator on a project of his, or Michael Tomasello, a scholar who challenged Chomsky's work. This thesis then visits the Wikipedia pages of other entities in the text with an emphasis on non-related entities if possible, therefore choosing to search first over Barbara Partee's page and Michael Tomasello's page to find genealogical relationships before visiting Elsie or David's pages, his immediate family members. This cycle repeats recursively until, be it time or random luck, a new search is started at a completely new entity, unrelated to any of the prior entities. This allows for documents in the corpus to be genealogically minimally related to each other and keeps us safe from traveling down group-specific rabbit holes that may introduce bias (such as gathering data on only the most prominent linguists). A visual representation of this process is shown in Figure 4.1, with green nodes being visited and red notes being found but not used for data.

This process creates documents in the corpus with entities from different social / professional circles (actors, linguists, politicians, athletes, etc.), as well as different genealogical family trees. Of course the process is not perfect, and some documents may well have entities who are related to entities of other documents, but the process greatly reduces this possibility.

Since this is a manual process, a list of entities is kept to keep track of entities whose Wikipedia pages have already been visited so as to not include duplicates.

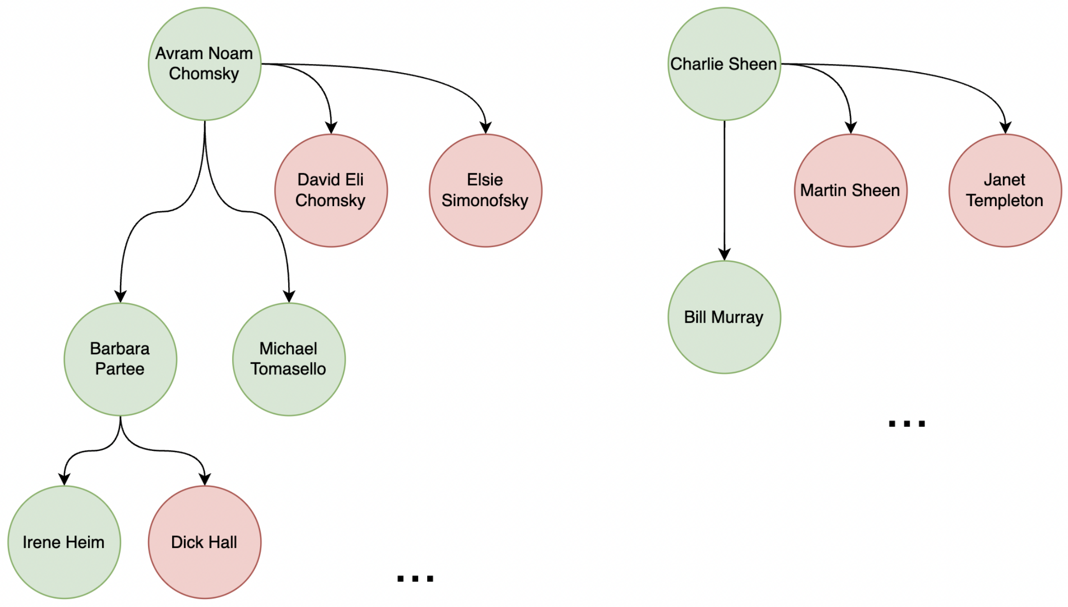


Figure 4.1: Examples of paths through Wikipedia one might take for document selection

Avram Noam Chomsky was born on December 7, 1928, in the East Oak Lane neighborhood of Philadelphia, Pennsylvania. [21] His parents, Ze'ev "William" Chomsky and Elsie Simonofsky, were Jewish immigrants. [22] William had fled the Russian Empire in 1913 to escape conscription and worked in Baltimore sweatshops and Hebrew elementary schools before attending university. [23]

Figure 4.2: Example unsanitized Wikipedia text with items to remove highlighted in red.

4.3 Document Preprocessing

Because pages visited are from Wikipedia, the sentences have a lot of citation syntax that is not typically present in natural language. An example is shown in Figure 4.2. Before being put into the database, the texts are sanitized for cases like this with regular expressions.

4.4 Paragraph Selection

Wikipedia pages, depending on the person entity they are about, can have an immense amount of data completely unrelated to genealogical relationships. While having some amount of non-relation-containing text is useful for building a system that can differentiate relevant relational data versus "fluff" text, each additional chunk of text with no relations only serves to make the manual annotation process more difficult. Thus, this thesis aims to choose a mix of paragraphs containing relations and not to minimize the cost of labeling while maintaining robust natural texts. The procedure for this is described below.

To begin, it is helpful to include the first paragraph of a Wikipedia text as it usually contains the full name of the entity that the article describes, which is useful for resolving mentions of that entity in later parts of the document. It is rare that this paragraph contains genealogical relations, but it is included nonetheless. Then, an annotator can read the page from top to bottom and if any paragraph contains a relation between any two entities (which does not have to include the page's principal entity), the entire paragraph is included in the data to be labeled from the Wikipedia page. If the paragraph is very long and the relation is found early on only, the paragraph can be shortened.

If a series of mostly consecutive paragraphs are found to have at least one relation each, but some paragraphs in between do not, these paragraphs should still be included in the document for the data set to be labeled. These intermediate paragraphs with no relations can contain relevant entity names for coreference resolution and relevant context, as well as provide additional data to the model that has no relations to increase its confidence in determining text that contains relevant genealogical clues versus not including such text.

In especially long documents that typically belong to famous person entities with dense amounts of historical information, it can be useful to have the main entity's WikiData page open on the side. Somewhat often the names of other entities that are genealogically related to them will be listed there in a key-value format, to which an annotator can just search the name on the page to find exactly which paragraphs contain the relation phrases. Though WikiData often does not have genealogically related entities, if it does it can be a nice time-saver compared to reading the entire bulk of a Wikipedia entry.

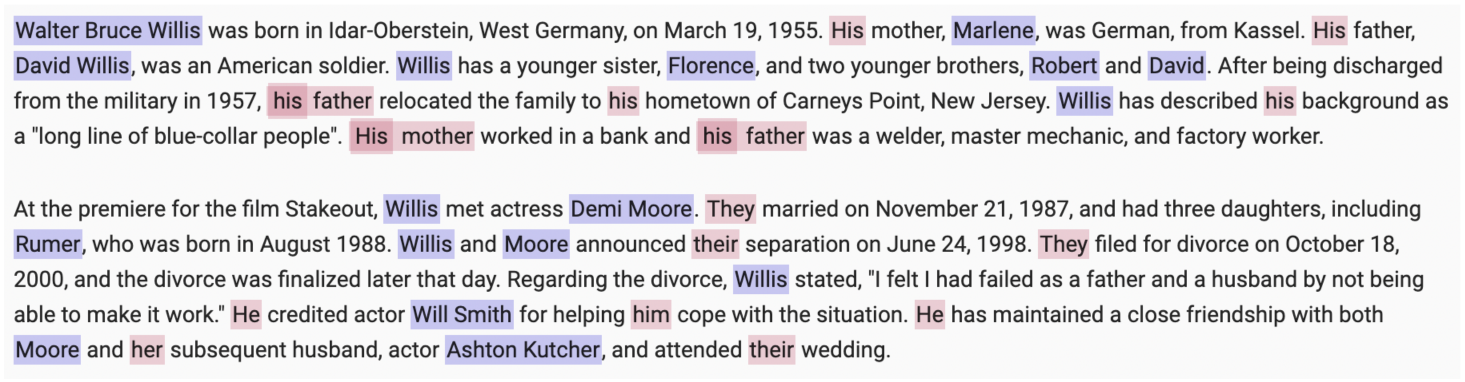
The paragraphs should be saved and compiled into a document as input to the DocPreProcessor module in the system, which is discussed in Chapter 5.2.

4.5 Labeling Entities

The labeling process is split into labeling two types of entities: named entities and anaphors.

This thesis defines "entities" as the proper noun strings / tokens that make up some or all of the name of a human reference in a document. The second type of entity, "anaphors" are any mentions to a prior named entity, whether it be a pronoun *or a noun phrase*. This distinction is important because, though many of the anaphors are pronouns, identifying noun phrases that resolve to an entity is crucial for finding relations within text between entities other than the principal entity of the document. Anaphors are linked back to the first name that they refer to, whether that be the most complete name or not.

In Label Studio, two entity types are made and a human annotator must label all proper nouns referring to a named human entity with the "entity" label, and all


Walter Bruce Willis was born in Idar-Oberstein, West Germany, on March 19, 1955. His mother, Marlene, was German, from Kassel. His father, David Willis, was an American soldier. Willis has a younger sister, Florence, and two younger brothers, Robert and David. After being discharged from the military in 1957, his father relocated the family to his hometown of Carneys Point, New Jersey. Willis has described his background as a "long line of blue-collar people". His mother worked in a bank and his father was a welder, master mechanic, and factory worker.

At the premiere for the film Stakeout, Willis met actress Demi Moore. They married on November 21, 1987, and had three daughters, including Rumer, who was born in August 1988. Willis and Moore announced their separation on June 24, 1998. They filed for divorce on October 18, 2000, and the divorce was finalized later that day. Regarding the divorce, Willis stated, "I felt I had failed as a father and a husband by not being able to make it work." He credited actor Will Smith for helping him cope with the situation. He has maintained a close friendship with both Moore and her subsequent husband, actor Ashton Kutcher, and attended their wedding.

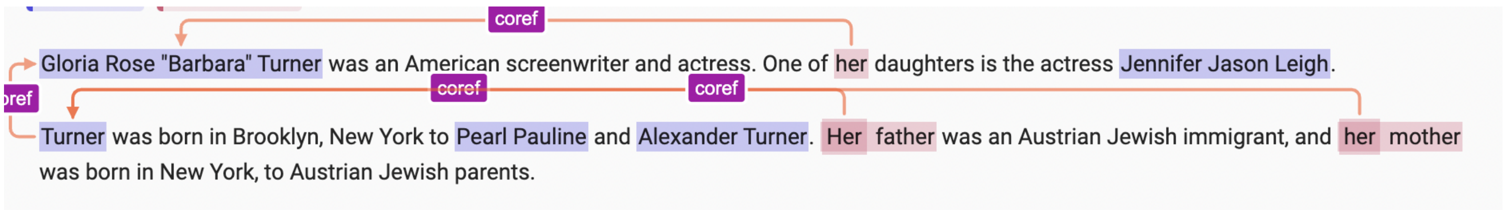
Figure 4.3: An example of labeled entities from Label Studio

pronouns, mentions, etc. (anaphors) as the second entity type so long as the entity they refer to is somewhere in the preceding text. An example of this is given in Figure 4.3, where named entities and their shortened proper noun versions are in blue and anaphors in red.

In cases such as the phrase “his mother“, there are two anaphor labels to be had, both “his” and “his mother”, since they both refer to separate human entities and both can possibly present a relation with other entities.

4.6 Labeling and Resolving Coreferences

Once all name entities and their anaphors are labeled, a "backwards" relation ($A \leftarrow B$) is to be made from all anaphors to their resolved entity, where A is the named entity and B is the anaphor. Due to the system that resolves coreference trees, discussed in Chapter 5.3.3, relations can be made from any anaphor to any preceding entity *or* anaphor so long as eventually those entities resolve back to the base entity. An example of this is shown in Figure 4.4. This can save time in labeling but the algorithm to resolve this in the end must be correct or else anaphors will resolve to different instances of the same named human entity. Furthermore, any pronouns or



Gloria Rose "Barbara" Turner was an American screenwriter and actress. One of her daughters is the actress Jennifer Jason Leigh.

Turner was born in Brooklyn, New York to Pearl Pauline and Alexander Turner. Her father was an Austrian Jewish immigrant, and her mother was born in New York, to Austrian Jewish parents.

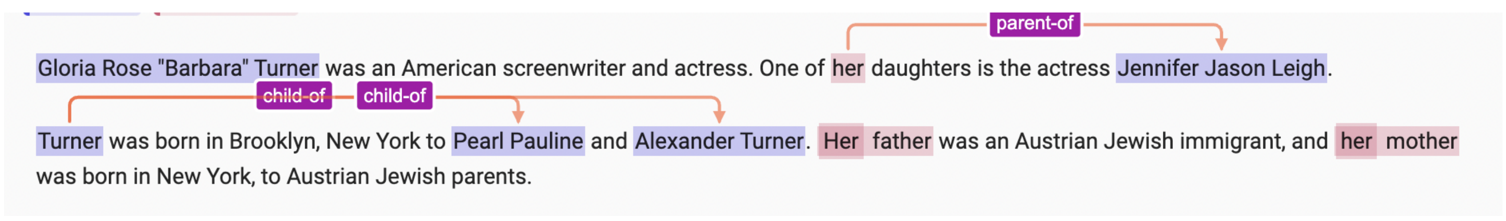
Figure 4.4: Labeled coreference relations in Label Studio

noun phrases that refer to multiple named entities must have backward relations to all entities they refer to. Since anaphors in text without the entities they resolve to should not be labeled in the first phase, they also don't need to be linked.

Though this step is not necessary for forming data necessary for the relation classifier, this is necessary for evaluating the performance of coreference resolution models, as this thesis relies on pretrained "out-of-the-box" models for coreference resolution. Each part of the system pipeline affects other parts, so this labeling step allows us to compare and interchange models depending on their domain-specific performances.

4.7 Labeling Relations

Once all entities are labeled with their resolved coreferences, the most important step follows: labeling relations within the text. A relation is formed only when two entities—either of type entity or anaphor—have a clear relationship as defined by the text between or surrounding the two entities. Additionally, the entities chosen to form a relation should be the nearest occurring entities to the context that reveals the relation, occasionally including entity mentions very close by. Furthermore, a relation should only be specified between two entities when the entities are the main subjects / noun phrases of the relationship. Some examples of this are shown in Figure 4.5, where sometimes the most descriptive anaphor is closest, sometimes the entity name.


Gloria Rose "Barbara" Turner was an American screenwriter and actress. One of her daughters is the actress Jennifer Jason Leigh.

Turner was born in Brooklyn, New York to Pearl Pauline and Alexander Turner. Her father was an Austrian Jewish immigrant, and her mother was born in New York, to Austrian Jewish parents.

Figure 4.5: Labeled genealogical relations in Label Studio

With a properly labeled data set containing all necessary information needed to produce data for training and evaluation, we can discuss the system design this thesis proposes in-depth. Conceptually, the pipeline follows similar steps to the labeling process to identify all necessary information for genealogical relationships. At the end, for training our models, it will make use of our hand-labeled data for fine-tuning.

Chapter 5

SYSTEM OVERVIEW AND DESIGN

The system contains many moving parts to communicate with labeling software, restructure data into DataFrames for use in training models on GPU-containing machines, and returning output relations in either knowledge graph (KG) or GEDCOM family tree formats. While each component will be touched on briefly, special attention will be paid to the DocRelations system, which does the bulk of the relation classification work using NER, a coreference resolution model, and a transformer-based classifier.

A view of the entire system and its components is shown in Figure 5.1

5.1 Entity and Relation Objects

The system contains two Python classes of importance, the `Entity` and `Relation` classes. These classes, and their instantiated Python objects, serve to convert strings in text to unique objects that can be identified later with an ID, start and end character token indices, and other metadata about the entity. It is not enough to store entities found in text as solely their string representations because there can be many occurrences of the same name within a text at different points in the text, and although they may refer to the same entity they likely have vastly different contexts around them and different surrounding entities that they may or may not be related to. Furthermore, this object allows for a universal interface to text entities returned by various systems (for example, Label Studio returns JSON objects, SpaCy returns Span objects, and so on).

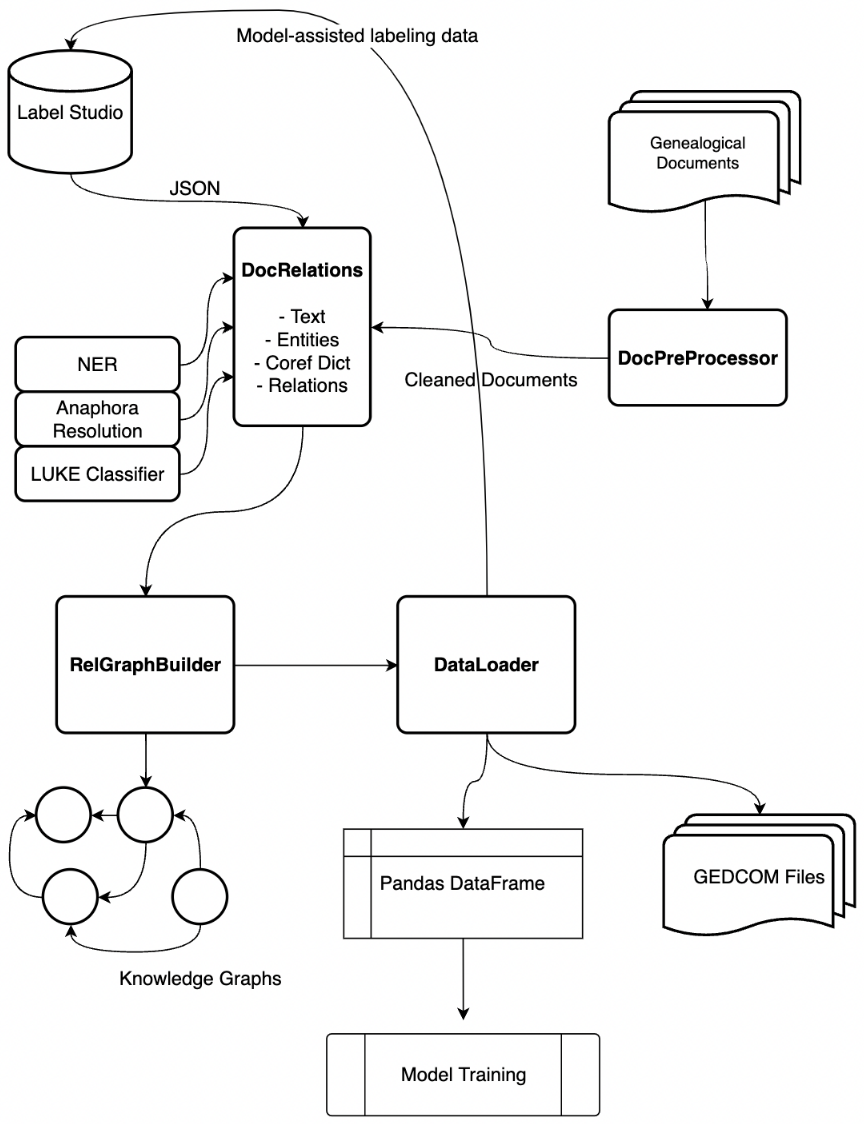


Figure 5.1: Entire system components and subcomponents

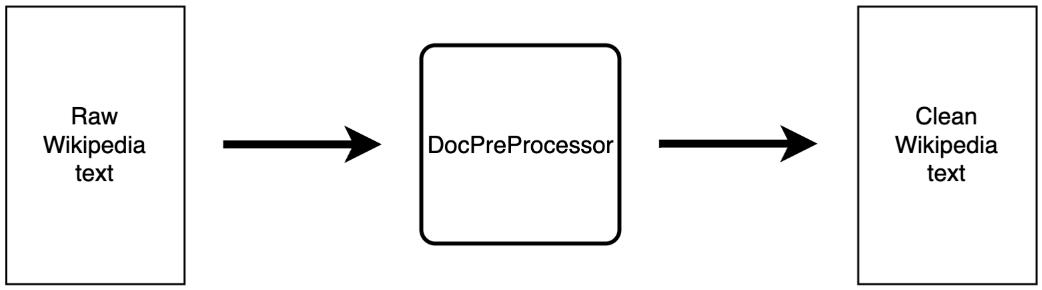


Figure 5.2: DocPreProcessor pipeline

The second class of importance is the `Relation`. These Python objects contain two abovementioned `Entity` objects, their relationship label, and confidence score for the relation if applicable. Having these objects allows fluid sharing of relational data between different parts of the system and creating relationship graphs at the end.

5.2 DocPreProcessor

One of the earliest parts of the system, the DocPreProcessor, is responsible for cleaning Wikipedia texts by removing citation syntax and other Wikipedia-specific data that might confuse the NER and coreference resolution models since these citations do not occur in natural language. It takes as input either a string denoting the entire document or a file to read and sanitizes it, seen in Figure 5.2.

5.3 DocRelations

This class, and corresponding Python objects, are responsible for the majority of the work in the system. A DocRelation object can be thought of conceptually as an object containing all necessary genealogical relationship information for entities within a single document. This includes storing information about entities, their anaphors, and their relations to other entities. The reason for loading this work on this part of the system is that genealogical relationship information can arrive either in Label Studio-specific JSON files denoting the "gold" relationship instances or must be inferred from text using models in the system. In either case, relations from the "gold" set must be equally and easily comparable to relations the system finds in text. This abstraction based on functionality also makes it easy to create training data to export to machines with GPUs for training relation classifiers. Finally, the `RelGraphBuilder`, discussed in a chapter below, can take a `DocRelations` object and easily parse it into a knowledge graph or GEDCOM file since all possible relations must be contained in a `DocRelations` object with their entities.

There are several models that this object uses for parsing documents, which are discussed below in chronological order.

5.3.1 Tokenizer

The first step that must occur in this system is the process of converting a multi-paragraph string into a list of "tokens." According to [28] tokens are the semantic units of processing formed from "chopping up" a string into pieces. Tokens can be punctuation, parts or wholes of words, and so on, are language-dependent, and are meant to provide systems with more semantically logical units to work with when compared to processing a string character by character, for example. While naive forms of tokenization can be splitting strings on whitespace or non-alphanumeric characters, modern-day libraries use statistical models for splitting text into more meaningful pieces.

The tokenizer this thesis uses is SpaCy's [7], which is widely known for being efficient, very accurate, and containing a plethora of additional grammatical and semantic properties per token, such as part of speech, shape, and lemma. Another popular library for tokenization is Python's Natural Language ToolKit (NLTK), though nowadays it is used less frequently due to its lower accuracy than SpaCy's and lack of additional semantic features. SpaCy's English model specifically is trained on OntoNotes 5, ClearNLP Constituent-to-Dependency Conversion (Emory University), and WordNet 3.0. It splits tokens in a statistical manner with some rules-based matching for exceptions per-language.

Once a document is tokenized in the first step of the pipeline for a `DocRelations` object, the tokens can be joined, filtered, and converted to `Entity` objects depending on their tokenized information.

5.3.2 Named Entity Recognizer

This is the first major step in the DocRelations pipeline that makes use of the tokens that were created earlier in the process. When SpaCy splits text into tokens, it attaches a host of information per token by running each span of text on a series of models. One of the labels it attaches is the named entity label per token if it exists. The definition of what makes up a named entity varies per dataset and domain, but SpaCy was trained on the OntoNotes 5.0 NER dataset, meaning that a named entity falls under 1 of 18 categories with the 4 most popular categories being geo-political entity (GPE), organization (ORG), location (LOC), and person (PERSON).

SpaCy learns what tags belong to what tokens by training a deep multi-task Convolutional Neural Network (CNN) on a multitude of tasks at the same time, specifically part-of-speech (POS) tagging, dependency parsing, and named entity recognition, which can all be found from the OntoNotes 5.0 gold dataset. The exact explanation of the training process and architecture of this NER model is hard to find, but it is noted in this video by a SpaCy developer that "Bloom" embeddings are used for representing tokens in this model. SpaCy reports results on the Named Entity Recognition task of the OntoNotes 5.0 dataset of 0.863 precision, 0.854 recall, and 0.859 F1 score, but it should be noted these are the scores across all 18 labels. Because this thesis is only concerned with extracting genealogical relationships from text, meaning they must relate two or more person entities, scores measured on this thesis' dataset are higher, likely due to the more predictable pattern of names than other entities.

Running SpaCy's NER system returns a list of SpaCy "span" objects that have metadata about spans of text within documents representing the names of person entities and their internal tokens, start and end locations, internal tags, etc. These spans are often made up of several tokens that were split earlier in the pipeline. An example could be Chomsky's father, Ze'ev "William" Chomsky, which could be represented as the list of tokens `[Ze, ', ev, ‘‘, William, '', Chomsky]`. This part of the system will only identify proper nouns denoting a person entity, meaning it performs best with full names but can catch both full names and nicknames for people entities. It will not return any tokens for representing anaphors, pronouns, or mentions of entities that are not proper.

The entity information is saved at this point in the pipeline and will be used in conjunction with the next step, coreference resolution, to find all possible entity names and mentions.

5.3.3 Coreference Resolver

The next part of the processing pipeline for a DocRelation object is performing coreference resolution. While relationships within text can be present between two named entities, namely something along the lines of "Adam was the brother of Maria," it is also abundantly the case where a *mention* of "Adam," such as a pronoun, is much

Adam loved going to the county fair, especially with his best friends Ethan and Tonya, so they decided to go after school. Since his sister, Maria, had never gone to a fair before, he decided to invite her.

Figure 5.3: A paragraph with a pronoun as the main entity for a genealogical relation

closer and more telling of a genealogical relation than where his name is. An example of this is given in Figure 5.3, where the entity of interest, Adam, is highlighted in red and the pronoun most telling of his relation to Maria is highlighted in blue.

Thus, if we had only detected and used named entities for finding relationships, we might have lower chances of resolving this relationship since there is much more context to parse, many more named entities in between, and a longer length of text between the two entities of interest. The goal of classifying relations in this thesis is to find and use the most useful contexts for defining a relationship, where minimal amount of extraneous details—“fluff”—are included.

To combat this, we can use a coreference resolution model to link all entity mentions to each other, specifically by linking them to their "antecedent," which is the entity they refer to (most often it precedes the mention).

In exploring coreference resolution models, this thesis looks towards using and evaluating two popular and high-performing models of interest, namely HuggingFace’s coreference resolution model “NeuralCoref” [37] [6] and AllenNLP’s coreference resolution model [1] [21]. Both of these models were evaluated based on compatibility with other system components, ease of use, and most importantly accuracy and performance on this thesis’ genealogical data set. Ultimately AllenNLP’s coreference resolution model was chosen due to its cutting-edge architecture and superior performance.

5.3.3.1 Output Processing and Usage

AllenNLP's coreference resolution model does not work straight out-of-the-box as needed, however. In order to get the best results for genealogical relation extraction, the output of the model must be filtered for use in the system. An overview of the entire algorithm is shown in Algorithm 1 and is explained in detail in the following chapters.

```
Input: 3-D array of coreference clusters with SpaCy-based entity indexing
Output: Sanitized coreference dictionary of entity keys and their mentions
        as values
populate_corefs(doc, coref_model):
    coref_dict = coref_model.get_predictions()
    for main_span, mentions ∈ coref_dict do
        names_POS = get_proper_nouns_POS(main_span)
        names_POS = cross_validate(doc.entities, names_POS)
        if multiple proper nouns and no possessives then
            split up entities and assign mentions
        end
        else if len(possesive nouns) == len(total nouns) then
            ignore span
        end
        else if len(possesive nouns) < total nouns then
            assign mentions to last entities mentioned
        end
    end
    for main_span, mentions ∈ coref_dict do
        for m ∈ mentions do
            if m ∈ coref_dict then
                new_ents = DFS_find_mentions(m, coref_dict)
                coref_dict[main_span].add(new_ents)
                delete coref_dict[m]
            end
        end
    end
    return coref_dict
```

Algorithm 1: populate_corefs: create a sanitized, genealogical-friendly coreference dictionary of entities and mentions from a coreference resolution model's predictions

First, we have to translate span representations based on token indices to `Entity` objects representing those same tokens. AllenNLP internally returns span clusters as a three-dimensional array of span ranges in the form

```
[
  [[start_idx_1, end_idx_1], [start_idx_2, end_idx_2], ...], # C1
  [[start_idx_1, end_idx_1], [start_idx_2, end_idx_2], ...], # C2
  ...
]
```

where values of the innermost length 2 arrays are inclusive start and end token indices that make up bounds of spans of the entities. These token indices are based on SpaCy's tokenization system, so in order to maintain consistency the `DocRelations` object ensures SpaCy versions are identical across the tokenization component and AllenNLP model. If this is ensured, entities and their mentions can be cross-compared with the original SpaCy document originally tokenized, and a dictionary of key-value pairs can be extracted, where the key is the antecedent or main entity and the values are a lists of mentions, pronouns, etc.

Secondly, additional steps are taken to filter out entities that are irrelevant or entities and mentions referring to non-person entities. Coreference resolution models are trained to recognize a multitude of entities and their various mention forms, though we are interested only in **person entities** and their anaphors. Furthermore, coreference models are trained to identify the entire span of antecedent mentions, so it might find a host of mentions for the entity "fellow basketball player Carmelo Anthony" instead of just "Carmelo Anthony," which is correct but includes unwanted preceding text that dilutes the purity of family trees.

{
Ze'ev "William" Chomsky : [his, their, they],
Elsie Simonofsky : [her, she, their, they], ...
}

Figure 5.4: Entities and anaphors where plural anaphors occur in multiple mention dictionaries

A naive approach initially taken was using SpaCy's NER parser again on the keys of the coreference dictionary to filter out all key-value pairs where the key (main entity) does not contain an entity. Though this works, it limits the ability to find named entities to the capacity of the NER model, which sometimes mistakes shortened versions of ambiguous names as other types of entities, such as locations or geo-political entities. A basic example of this could be someone named "Hannah Montana," where it would not be uncommon to refer to this entity later in the passage as "Montana," a reasonable location confusion a model might make. Thus, the approach this thesis takes is to go through every entity span key in the coreference dictionary and parse through the part-of-speech tags of every token, isolating proper nouns and taking note of if they are possessive of something in the sentence or not. All found spans of proper nouns are then cross-referenced with entities from the NER model and if any entity from the NER model or any proper noun from the coreference model spans are string-wise subsets of each other, then they are kept as entity mentions. Thus, so long as the NER model picks up "Hannah Montana" as a named person entity and the coreference model picks up all references to them, with confusing references such as "Montana" (a location by NER model's view), then they will all be kept as valid mentions. Furthermore, this cross-checking prevents against keeping most other proper nouns (locations, organizations, etc.) such as "NBA," which is highly unlikely to be a subset of a named person entity found by the NER model.

Antecedent entity spans might also contain multiple entities where mentions can be of the form “they,” or “their,” in which case we separate the span key into multiple new keys each with unique entities from the span containing just their proper names. An example of this is given in Figure 5.4, where two entities in a span both get the same mention values but separate entries in the coreference dictionary. In this step, caution is taken as oftentimes a span contains one or more proper names for entities but is only referring to a subset of the names or none at all, depending on the context. One example for this is the span “Michael’s summer vacation,” which the coreference model might correctly link with further mentions of summer vacation such as “the trip” or “the tropical vacation.” It would be incorrect to include mentions similar to “the trip” as mentions to “Michael,” which we would find after parsing. Furthermore, in a span of text like “Michal’s sister, Serena,” with mentions “her” and “his sister,” it would be incorrect to link those mentions to both Michael and Serena since they only truly refer to Serena. For such cases, this system looks at proper noun sequences, punctuation, and the beforementioned possessive property of tokens. If entity spans contain a possessive entity with no other entities after, then the span is discarded. If there is an entity after a possessive entity, then only the last entity is kept to link to the mentions.

This rule is also roughly translated to the mention spans from the coreference model. For some reason, if a noun is possessive like “Martin’s”, AllenNLP’s coreference resolution model will include the entire span instead of just “Martin.” This happened many times over the training set, so if any mention spans ended in a possessive token like “ ’s ,” it was removed as it doesn’t make grammatical sense. This is the only rule applied to anaphors or mentions of entities because we cannot make guarantees about these tokens as there are many ways to refer to an antecedent (many ways to mix pronouns, not easily parseable like proper nouns are). Creating so many grammar parsing rules also defeats the purpose of a coreference resolution model that should learn to extract entity mentions correctly without needing a set of NLP rules.

Finally, after filtering is done and the coreference dictionary is simplified to key-value pairs where the keys are solely proper names referring to entities without unwanted surrounding text, the coreference dictionary is collapsed and combined such that all mentions to the same entity key in the dictionary are in its values. The coreference resolution model sometimes finds different, disjoint mentions of the same entity but fails to reconcile them under one antecedent. This step involves iterating over all entity mentions and recursively "shifting" them over to their correct antecedent. An example of this is given textually in Figure 5.6, which can be visualized graphically in Figure 5.5.

This thesis *does not* replace anaphors or mentions in text with their antecedents and instead opts to save this information in a dictionary for future use. This is because this thesis aims to preserve text as much as possible before performing relationship classification so that word embeddings are contextually accurate, which is explained more in Chapter 5.3.4.

5.3.4 LUKE Entity Pair Relation Classifier

While the prior two steps are important pieces of the pipeline for a `DocRelations` Python object, they cannot accomplish relation classification without this crucial component of the thesis. From the related works chapter, we can see that the field of genealogical relation extraction suffers from a few problems, making relation classification and choosing a fitting model for relation classification (RC) a hard task. The BERT-based transformer model that was used, LUKE, solves these problems.

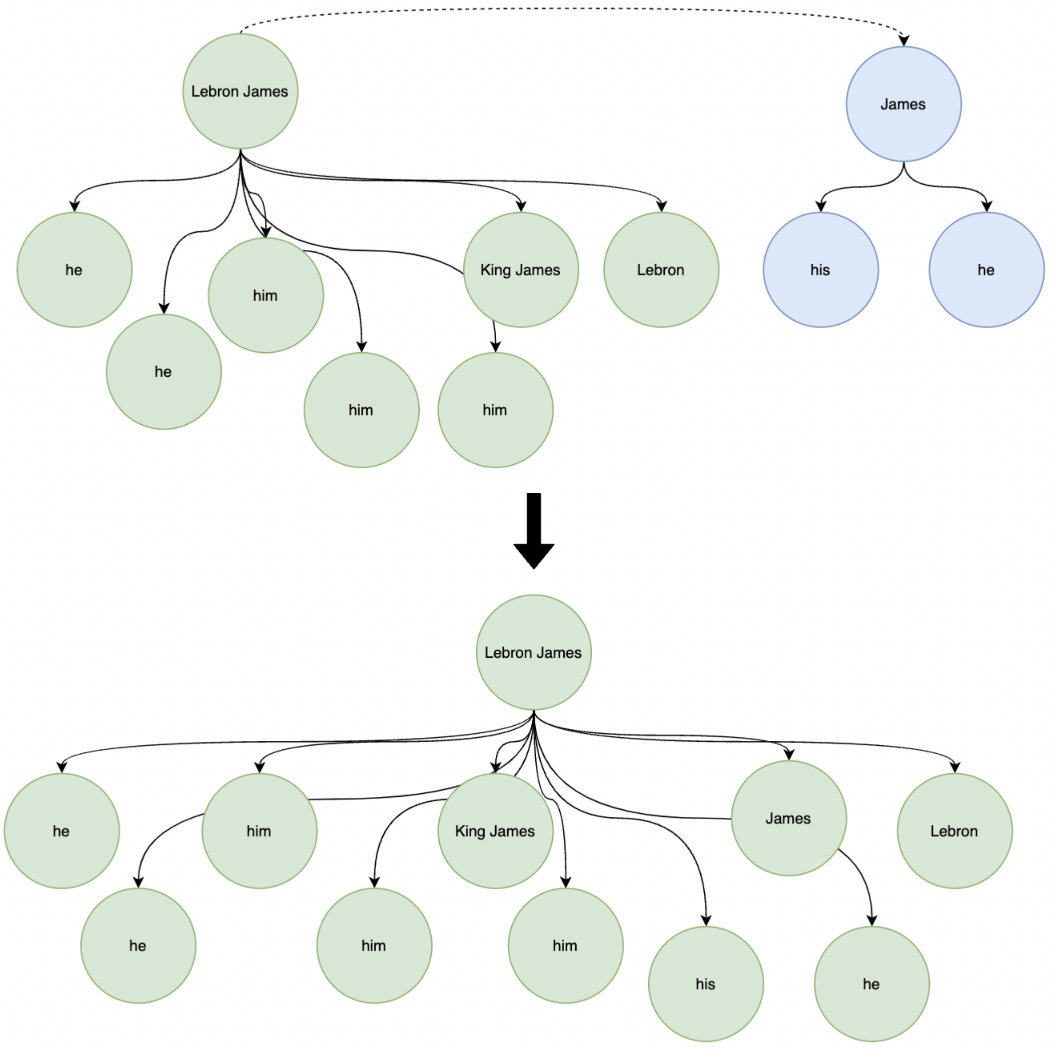


Figure 5.5: Combining coreference dictionries into acyclic trees

{
Lebron James : [he, him, King James, Lebron, he, James, him, him],
James : [his, he], ...
}

$\rightarrow$

{
Lebron James : [he, him, King James, Lebron, he, James, his, he, him, him],
}

Figure 5.6: Coreference dictionary entries that should be unified

Most notably, there is no unified data set for golden-labeled genealogical data, nor are there rules or guidelines denoting how labeling should be done, creating for a very data-scarce environment. This has a profound impact on model selection for classifiers as prior work shows that rules-based approaches do not take too well to unstructured texts and statistical models that utilize machine learning are extremely data-hungry. Deep learning models, which are increasingly setting new state-of-the-art scores across many NLP tasks, require even more data—typically on the order of tens of thousands of samples at the minimum for decent performance.

Another issue present across genealogical relation extraction is, when using a machine learning method, how to choose features. Current methods mainly use hidden Markov models, support vector machines, or naive Bayes classifiers, amongst others, alongside a plethora of hand-crafted features, such as distance between entities, word count, presence of key tokens, and so on. There is no exhaustive list of features that should or should not be included, and combinations and choice of features have profound effects on classification results. Furthermore, representing semantic and syntactic knowledge of text through these features and simple vectorized representations, such as word count (bag of words) or TF-IDF, fail to instill deeper knowledge into these deep learning models about contextual representation of words or spans.

Finally, the choice of how to select candidate entity pairs for relation classification depends on the problem definition. In our case where we want to resolve relations for any entities, this problem becomes more complex.

In order to solve the issues of data scarcity and feature selection, this thesis takes the approach of fine-tuning a massively pretrained BERT-based model, LUKE, while preferring minimal features over heavily-featured approaches, a trend identified by [22]. This approach is possible thanks to the deep understanding of syntax and semantics from LUKE's self-supervised pretraining tasks on massive corpora. For generating candidate pairs, a sliding window approach is used to classify entities or anaphors near each other.

5.3.4.1 Generating Entity Pairs

Before relations can be classified between entities, this thesis starts by building candidate entity pairs for classification. This step is only possible because of the prior steps of identifying all entities, their mentions, and resolving mentions to their original antecedents is done. Within the `DocRelations` object, two parameters, namely `max_span_len` and `edge_chars`, can be specified by a user to determine the entity pairs to generate. This thesis uses a sliding window approach to pair potentially related entities, meaning that it iterates over the entities and mentions found in the prior components of the pipeline creating empty `Relation` objects with them if they are within `max_span_len` distance from each other. This thesis defines "within" as the first character of the earliest entity mention being $\leq$ `max_span_len` characters away from the last character of the latest entity mention. Furthermore, for a valid relation to be created, this thesis utilizes the coreference dictionary and `Entity` objects to only create relations where entities do not overlap and where two entity mentions do not refer to the same coreference resolved entity. The entities are not modified in any way and names are not replaced, as replacing text with proper nouns might modify how token vectors are calculated and our goal is to keep text as pure as possible to fully utilize pretrained embeddings.

The choice of `max_span_len` chosen for this thesis was 160 characters. This was found by looking at the distribution of relations from the golden labeled data set and choosing a span that captures 99% of possible relations from the training set (as choosing these on the test set introduces bias). This accounts for some relations being outliers while not sacrificing the ability to identify most relations. Increasing the

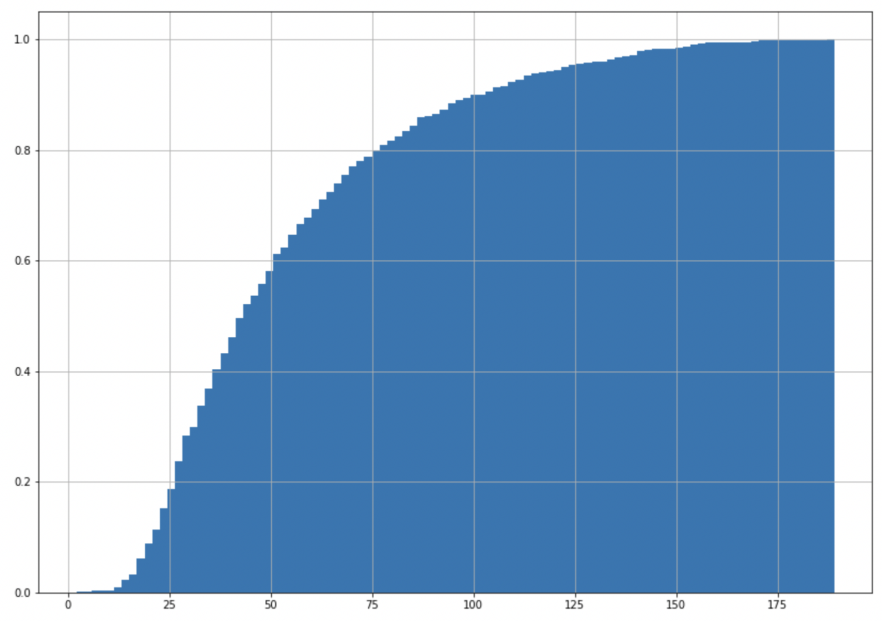


Figure 5.7: Cumulative histogram of span length vs. % data set covered

`max_span_len` heightens the possibility of identifying negative relationships, so finding a balance that captures most relationships while reducing effective `max_span_len` is essential. A cumulative histogram can be seen in Figure 5.7, with span length in bins on the x-axis and percent of data set covered on the y-axis.

Once all possible combinations of entities within proper distance of each other are found, a Pandas DataFrame is created (using the `DataLoader` object) of these relations by including the text between them as well as `edge_chars` characters both before the first entity mention and after the second entity mention. This is done because it is sometimes the case that the necessary context for two entities to be related is not located between them, as can be seen in Figure 5.8, where related entities are highlighted in green and the context needed to define the relationship is highlighted in blue. Furthermore, additional context can help with relations that have little text in between. Added to this DataFrame are the relative locations of the entity spans for each entity based on character indexing, as is required for LUKE's tokenizer to properly represent the entities.

Adam was on the set for a new movie when he met his co-star Evelyn.
The two married a year after the movie was released.

Figure 5.8: A relationship where context defining the relation is found outside of the text between two entities.

Hyperparameter	Values
Epochs	`[1, 2, 3, 4]`
Edge_chars	`[15, 30, 45, 60]`
Negative_rels	`[0.25, 0.50, 0.75, full(∼0.90)]`

Table 5.1: Possible hyperparameter values

5.3.4.2 Model Architecture and Hyperparameters

We use the default cross entropy loss provided by Pytorch, which takes in raw logits from the model output layer and applies the $log(Softmax(x))$ for each tensor and calculates the negative log likelihood loss. The AdamW optimizer is used, an optimizer that adjusts gradients based on the moving averages of the gradient [25]. There are 5 possible labels for our model, namely "parent-of," "child-of," "sibling-of," "spouse-of," and "no-rel." We fix the learning rate at $2e^{-5}$ and perform hyperparameter tuning on the parameters and values seen in Table 5.1 using a grid search, where the best model is selected and used moving forward. We found that `edge_chars=45, epochs=4, negative_rels=full` gave the best performing model. This hyperparameter tuning is discussed more in Chapter 6.4.1.

5.3.4.3 Training

Entity pairs for training the classifier are similar but different from the beforementioned procedure. The sliding window approach is still utilized, but only for negative relationships as the golden labeled data already contains all known relations. Therefore, known relationships are filled first with `Relation` objects and corresponding

`Entity` objects containing their true relations. All other entities within their respective window lengths to other entities are then built as negative "no-rel" `Relation` objects. All possible negative relations are contained in a `DocRelations` object, and the negative relations can be undersampled as needed in hyperparameter tuning in the future.

For training the model, we make use of PyTorch's `LightningModule`, `DataSet`, and `DataLoader` classes to construct PyTorch objects that can be trained with the architectures given in Chapter 5.3.4. The model was trained on a Google Colab Pro instance using one NVIDIA Tesla P100 GPU. Batches from the `DataLoader` objects are of size four and samples are shuffled during training to avoid training on consecutive samples from the same document. For any random behavior or shuffling, a random seed is used for reproducibility.

5.3.4.4 Classification

Classification functions similarly to training in which PyTorch's `LightningModule`, `DataSet`, and `DataLoader` classes are used, though in this case shuffling is turned off and classifications are made directly on the DataFrame built from building entity-pairs. Once relations are created, `Entity` and `Relation` objects can be reconstructed from the new DataFrame with their original entities. When reconstructing relations, all relations classified as "no-rel" are discarded and any predicted relations with confidence scores less than 0.40 are discarded as well. The 0.40 threshold was chosen to remove very low confidence predictions as we are trying to build a high-precision system. 0.40 represents confidence higher than a random guess (0.20).

5.4 RelGraphBuilder

This component of the system consumes a `DocRelations` object and builds a genealogical family tree knowledge graph. This is the purest, final form expressing genealogical relations in this thesis, where related nodes in the graph are resolved to their most telling proper names. The `RelGraphBuilder` object iterates over all of the relations in a `DocRelations` object and uses a reversed coreference dictionary (where keys are now mentions / anaphors and values are their antecedents) to resolve all of the entities in the relations. If there is a relation present between two entities, the respective entity nodes are added with their metadata in the node. They are then connected by a directed edge in the graph in the same order the `Relation` object defines the relationship. If either entity in the relation some mention like "they" that could refer to multiple people, edges will be added from each entity to the other entity they are related to after resolving the mention.

Though uncommon, it is possible that our model classifies multiple relations for the same sets of entities. Since this is genealogical relation extraction, an entity can only have one relation edge to another entity (you cannot be someone's sibling and parent, for example). Thus, if the system encounters this during parsing, then the confidence score is checked for the relation label, and if it it is higher than a previously assigned relation all old relation edges are replaced with new relation edges with the new label. Essentially what this means is that the model has found some context later in the text between the same two people that is less ambiguous and offers better insight into the relationship between two entities.

This component of the module uses Pyvis as the graph library of choice for its ability to create interactive graphs by acting as a wrapper around a popular JavaScript graph library. It renders HTML pages with interactive nodes, making it useful for longer documents with more relations, where popular Python graph packages, such as NetworkX hide relations due to it creating static images.

5.5 DataLoader

The `DataLoader` is an intermediate component between parts of the system, that is mainly concerned with packaging data for use in various other systems. It has three primary functions that all consume a `DocRelations` object.

1. Creating JSON data for labeling - The process of labeling can be sped-up by using models to predict entities and relations before having a human do it. Once a model achieves reasonable accuracy, it is faster to have a human perform labeling by just checking validity of model-made predictions versus starting from scratch. The `DataLoader` is capable of taking in a `DocRelations` object and creating a JSON file for use in Label Studio as a document with predicted annotations.

2. Creating Pandas DataFrames for training and classification - In many instances, when training a model, it is almost a necessity to use a GPU (for example, so that fine-tuning takes less than a day). This functionality allows the creation of pickled DataFrames that can be exported from some local machine and sent to a GPU-enabled machine like a Colab instance. These DataFrames consume DocRelation objects and build up relevant information, such as the base text between and around entities, entity locations in the span of text, locations in the original document, and so on. They contain all information needed to reconstruct `Entity` and `Relation` objects before and after classification.

3. Creating GEDCOM family tree files - To make the overall system useful to genealogy companies, this component of the module consumes a `RelGraphBuilder` object and returns a GEDCOM file specifying hierarchical family trees that can be used in other programs. It consumes a graph instead of a `DocRelations` object because a `RelGraphBuilder` has already resolved entity mentions and constructs a genealogical knowledge graph.

In order to evaluate the efficacy of the design choices and model choices this thesis makes, we must perform experiments. When designing experiments, we should aim to use methods that give us the most unbiased, truly indicative measures of performance in the real world. Additionally, because our pipeline has so many different communicating pieces, we must evaluate the system as a whole alongside its functioning pieces.

Chapter 6

EXPERIMENTS AND MODEL SELECTION

This thesis proposes a three-part pipeline with intermediate objects for communication that is consistent with other literature. While most literature chooses to focus on reporting results for trained classifiers that have been used, a main focus of this thesis is evaluating and identifying weaknesses across all components of the system alongside evaluating the classifier. These other components and their evaluation within the genealogical domain specifically are not often discussed and can present important information for the future of related work. Our evaluation methods also show distinctions between models evaluated on standard data sets and models evaluated on genealogical texts that necessitate this discussion.

Additionally, this thesis draws a distinction between the evaluation of each component individually and the evaluation of the full end-to-end pipeline. A novel method for evaluating the full pipeline is described and used to compare the current system's performance against theoretical best performances when fixing certain aspects of the system. This is all possible due to the comprehensive, golden-labeled data set.

The remainder of this chapter discusses how the NER, coreference resolution, and classification models are evaluated. Furthermore, the hyperparameter tuning and end-to-end evaluation methods are discussed.

$$Accuracy = \frac{\#\ labels\ correct}{\#\ labels\ total}$$

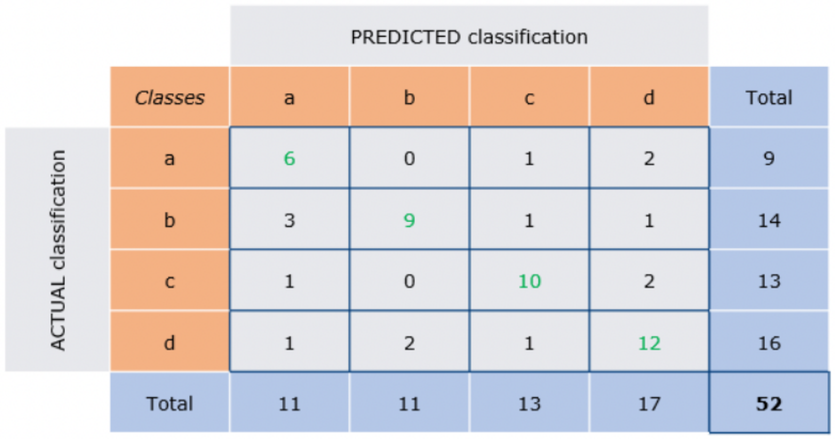
Figure 6.1: Accuracy formula

6.1 Useful Metrics for Classification

Choosing a metric for evaluating classification models can greatly impact results and make the difference between numbers that display over-biased estimates and numbers that represent a model's true performance. A very common and simple metric for evaluating models is accuracy, where the score is simply a function of labels assigned correctly and incorrectly. The formula is given in Figure 6.1.

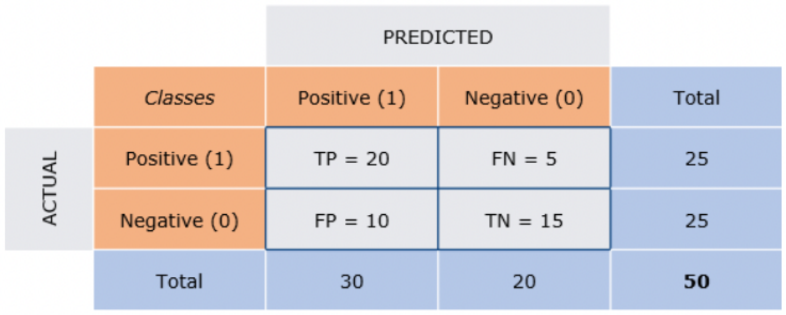
While the metric gives some insight into performance of a model, in multiclass classification tasks, such as the one this thesis tackles where there can be several possible labels of vary distributions, this metric is inadequate. There are two main reasons for this. The first reason is that accuracy scores give no insight into which labels the model performs better or worse on. Since accuracy is simply a function of exact matches, we have no clue which labels are contributing to the success or failure of a model. Secondly, in classification tasks accuracy can lead to a false sense of security with labels of differing distributions sizes. For example, if you have a binary classification task for credit card fraud, where potentially 98% of samples are not fraudulent (negative) and 2% that are (positive), and your model achieves an accuracy of 98%, then you may think it performs very well. In reality, to achieve this score the model can simply always output non-fraudulent label and miss every label of importance—the fraudulent cases.

For these reasons, classification tasks usually use the metrics of Precision, Recall, and F1 scores, which utilize the commonly used confusion matrix. According to [16], a confusion matrix is a "cross table that records the number of occurrences between two



		PREDICTED classification				
	Classes	a	b	c	d	Total
ACTUAL classification	a	6	0	1	2	9
	b	3	9	1	1	14
	c	1	0	10	2	13
	d	1	2	1	12	16
	Total	11	11	13	17	**52**

Figure 6.2: An example confusion matrix of multiple classes



		PREDICTED		
	Classes	Positive (1)	Negative (0)	Total
ACTUAL	Positive (1)	TP = 20	FN = 5	25
	Negative (0)	FP = 10	TN = 15	25
	Total	30	20	**50**

Figure 6.3: An example binary classifier confusion matrix

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 * (\frac{precision * recall}{precision + recall})$$

Figure 6.4: Precision, Recall, and F1 formulas

raters, the true/actual classification and the predicted classification." The figure they show is seen at Figure 6.2. The most simple form of a confusion matrix is seen in Figure 6.3 with a binary classification class, where there are only two classes possible, a positive and negative class. These confusion matrices are based on the number of true positives, false positives, true negatives, and false negatives. A true positive is when the model predicts a sample to be of the positive label and the actual label of the sample is positive. A false positive is the inverse, where a model predicts a sample to be positive when in reality it is negative. These same rules apply to negative examples. Using these metrics, we can define precision, recall, and F1 scores. [16] defines precision as "the fraction of true positive elements divided by the number of positively predicted units." Conceptually, this metric is a score that tell us "how much we can trust the model when it predicts an individual as positive." Recall, on the other hand, is defined as "the fraction of true positive elements divided by the total number of positively classified units," which we can think of as the true amount of samples that are actually positive. This, therefore is a measure of how well a model finds all possible positive examples in the data. Models with high precision and low recall, might not find every case of a positive relation but will be very trustworthy. Models with high recall and low precision find many of the true positive examples, but might mistake several examples to be positive instead of negative. The F1 score is the harmonic mean of the two metrics, giving us an idea of how they play together. The formulas for precision, recall, and F1 score are shown in Figure 6.4.

While calculating precision, recall, and F1 scores are simple on binary classification tasks, for multiclass classification when there are multiple "positive" and "negative" labels this must be calculated per-label and averaged for a comprehensive score. For averaging methods we can either use the "macro" precision, recall, and F1 scores or "micro" precision, recall, and F1 scores. In macro scores, the metrics are calculated per each label's true/false positives and negatives and averaged at the end where each label's score is weighted equally. In the micro scores, true/false positives and negatives are calculated for each label and summed over all labels and then averaged, essentially producing a "weighted" average of metric scores where majority classes have much more influence. For this thesis, the "macro" weighted scores will be chosen when possible because they give a true indication of how the model is performing across all labels and is not influenced by heavy data distribution. This is especially relevant in genealogical relation extraction because, when constructing candidate entity pairs in documents, there is a very high presence of unrelated entities, especially in entity-rich documents. While classifying them as having no relation is important, we care more about about how the system identifies positive relation classification with no special treatment to the universal negative class "no-rel."

6.2 Evaluating Entity Detection Metrics

Evaluating the NER model, since there is only the presence of names or lack thereof, follows precision, recall, and F1 scores of a binary classification. To begin, two sets of DocRelation objects are made for each document in our data set. The first set of DocRelation objects are instantiated from the gold-labeled data set where proper noun entities, anaphors, and relations are correct. The second set of objects utilize the three-part pipeline to identify entities, anaphors, and relations. We can utilize the entire data set and not have to perform a train-test split because this thesis uses a pretrained NER model and performs no additional training. Thus, it has not seen any data in our data set. Once done, two experiments are run.

The first experiment seeks to validate the performance of solely the named entity recognition model. The NER model is ran on all of the documents without adding any anaphors, only finding full names or nicknames of entities in text. The gold-labeled data has a distinction between entity names and anaphors as well, so they can be filtered out. Once done, two sets are made for `Entity` objects found by the model, which we will refer to as set A, and those found from the gold data, set B. The intersection $A \cap B$ of the two sets are entity names that the model found which truly exist in the gold data. These are the true positives in our entity scoring. The difference $A - B$, meaning all entities the model found that do not exist in the gold data, are false positives. The difference $B - A$, all entities in the gold set that were not found by the model, are false negatives. Using these values, precision, recall, and F1 scores are calculated.

The second experiment seeks to validate the performance of finding all entities and anaphors, without attention to which antecedents any of the anaphors refer to. This is meant to assess the ability of the system to generate all possible entity and entity mention combinations. For this, entities found by the NER model and coreference resolution model are cross-referenced as explained in the System Overview chapter and compared against all entities and anaphors in the gold data. Sets are created equally to the first expirment and metrics are calculated in the same way.

It should be noted that for any set operations, two `Entity` objects are only considered equal (intersect is true) if and only if their text string, starting character index, and end character index match up with each other and their respective locations from the original documents. This is because there can be many references to the same entity meaning lots of duplicate strings in the text. If `Entity` objects were evaluated on text alone, finding one instance of a name would be equivalent to finding all instances of a name, which is incorrect. As an example, the object `Entity(‘‘Lebron James’’, 3, 15)` would not be equivalent to the object `Entity(‘‘Lebron James’’, 43, 55)` because they are separate instances that occur in different parts of the same document.

Scores are calculated per-document and averaged, giving us a per-document average precision, recall, and F1 score, and true/false positives and negatives are summed across all documents before averaging as well, giving us an idea for how many total entities are found. This is roughly the equivalent of “macro” and “micro” averages, where macro scores are averaged over each document and micro scores are using data from all documents as one.

6.3 Evaluating Coreference Resolution Model

This experiment seeks to validate the performance of the coreference resolution model and its ability to resolve anaphors to their antecedents. Similar to the NER evaluation, since there is no training and we are using a pretrained model, our entire data set can be used for evaluation. It follows a very similar pattern to the experiments for entities in the prior chapter with adjustments due to the nature of coreference resolution. `DocRelation` objects are created as before using gold data or model inference, where the resolved coreference dictionary in either case is a dictionary of key-value pairs in which each key, an `Entity` objects, has as its values a list `Entity` objects of all anaphors that refer to it. Thus, before we can split values into sets we iterate over all keys and values in the dictionaries and create tuples of size two, where the 0th index is antecedent entity and the 1st index is the anaphor that it refers to. Once tuples are created from all resolved entity mentions, they can be compared in the set fashion described in the previous chapter. Precision, recall, and F1 scores are measured in the same fashion with the difference being that in order for two tuples to be the same, the `Entity` objects in **both** indices must match with each other and must be in the same location.

Initially, because coreference models sometimes grab additional context surrounding an entity, partial scores were given depending on the percentage each entity was represented in either tuple. This was discarded, however, as coreference dictionaries have already been parsed to remove extraneous context as described in the System Overview and the best scoring future coreference models should learn to include only meaningful tokens that refer to entities and mentions. By using a partial scoring system and allowing extraneous information to be included with entities and mentions, weaker coreference resolution models that choose to predict longer spans will unjustly be rewarded. In the worst case, a coreference model that predicts the span containing the entire document every time will technically capture any relevant entity.

Like in the NER evaluation, scores are averaged per-document and summed before averaging, giving us a notion of macro and micro average precision, recall, and F1 scores. This experiment is ran for HuggingFace's NeuralCoref resolution model and AllenNLP's coreference resolution model.

6.4 Fine-Tuning and Evaluating LUKE model

6.4.1 Hyperparameter Tuning

Before a classifier model can be evaluated, we must first evaluate the combination of hyperparameters that give the best final model. Possible hyperparameters and their values are shown in Table 5.1. `Epochs` are the amount of epochs trained over the training set where an epoch is a full iteration over every sample of a set. `Edge_chars` are the amount of characters included as context both before the first entity and after the last when classifying a relation and feeding text to a LUKE classifier. `Negative_rels` is a variable controlling the undersampling of negative relations in the training data set. The value for `Negative_rels` describes the distribution of negative relations in the training set. For example, the value 0.50 implies a distribution where the amount of negative, “no-rel” samples in the training set makes up 50% of all samples, where the sum of all other classes make up the other 50%.

In classification tasks where there is a heavy class imbalance (as in this thesis, where negative relations form roughly 90% of all relations in a document), undersampling or oversampling can be used to aid models learn class representations better. With undersampling, some fraction of samples from the majority class(es) are removed from the training set to not overwhelm the other labels and even out the distribution of class samples. Oversampling makes use of data augmentation techniques to increase the amount of samples from minority classes. Both techniques have their dangers, such as removing useful examples from majority classes or introducing noise to minority classes through data augmentation. Furthermore, a fundamental principle in training statistical models is that they should use data representative of real-world population data as the predictions made during use will follow the distribution learned turing training. Thus, undersampling and oversampling likely skew the distribution of predictions on real world data.

Oversampling was not chosen as a valid technique for this thesis because we want to take care to train the model on highly accurate golden relations and not create additional noise in generating samples that can bring precision down. Furthermore, with no guaranteed structure to genealogical relations and no guarantee of any predictor tokens like “sibling” or something of the sort, data augmentation becomes a difficult issue. Undersampling is tested, however, in the hopes that negative relations have similar representations compared to positive relations and that including less and allowing the model to better learn positive relations might potentially outweigh the effect of changing the distribution of training data.

6.4.1.1 Avoiding Bias and Leakage in Evaluation

Before any evaluations are done for our classifier model, our data set of 104 labeled Wikipedia documents, consisting of 777 relations, is split into two sets: a training set and a testing set. The split distribution is a 0.65 - 0.35 training-testing split. The split is not done over the pure number of relation samples in the data set, as this can introduce bias into our model. For example, we might have relations from the same families mixed in the training set and testing set. A model can then learn to overfit on the data by predicting based on entities in sentences rather than the context surrounding them. Due to our labeling process greatly reducing the familial overlap across documents, we split our training and testing set based on document IDs, meaning each set has relations from different documents where no single document or any relations within appear in both training and testing sets.

Furthermore, while 0.80-0.20 training-testing splits are more common in literature, this thesis chooses 0.65-0.35 thanks to the benefits of using pretrained transformers. As the number of relations are small due to the cost of labeling and time constraints for this thesis, the testing dataset has to have a sufficient amount of samples per relation class to predict on to reduce variability in calculating final scores. Having a larger testing split allows for this, but leaves less data for training and hyperparameter tuning. The pretraining of transformers, however, result in baked-in contextual weights that require minimal extra data to fine-tune. Thus, having a smaller training set, this thesis hypothesizes, is minimally hindering to end results. This split allows for roughly 50-70 examples in each positive class and many negative examples to be evaluated, a fair amount for proving a proof-of-concept. The distribution of classes is shown in Table 6.1.

Label	# Relations
No-rel	2,141
Child-of	71
Parent-of	77
Sibling-of	57
Spouse-of	72

Table 6.1: Relation distribution in testing set

With this 0.65-0.35 training-testing split, the model never sees documents and relations from the testing set in any hyperparameter tuning, and no judgments can be made based on the testing set for any choices of model parameters. The testing set is strictly for reporting performance on theoretical "real-world" data, and all performance and hyperparameter decisions are made solely on the training set. In all, this makes for a rigorous, robust testing method where the absolute minimum, if any, bias or leakage is introduced.

6.4.1.2 10-fold Cross Validation

In order to perform hyperparameter tuning to find the best parameters for the data set, while using a reduced training set and not making predictions on the testing set, k-fold cross validation was used where $k = 10$. 10 folds for cross-validation is standard in literature as a number that allows for sufficient amounts of data to be used per fold while keeping computation costs low.

In 10-fold cross-validation, the training set is partitioned into 10 separate chunks where, for 10 iterations, a newly initialized model is trained on $k - 1$ chunks, in our case 9 which we will call the new training set, and the trained model is evaluated on the held out chunk, known now as the development set. Repeating this process 10 times, 1 for each fold we create, allows us to train models on 90% of the training data at any one time and evaluate on the entire training data set after 10 iterations. For each fold, we calculate true/false positives and negatives and at the end calculate macro and micro averaged precision, recall, and F1 scores for all of the development sets.

It's important not to calculate precision, recall, and F1 scores for each label for each development set too early. If calculated for each fold and averaged weighting each fold equally, folds with lower samples and thus higher variability in score can impact results across the rest of the folds. Thus, only the components of these scores, true/false positives and negatives are calculated and summed up after all 10 folds have run to get a true metric for macro and micro precision, recall, and F1 scores.

For calculating these metrics, now we must truly calculate true and false positives and negatives per each label and take macro averages, which can be done easily using Sci-Kit Learn's confusion matrix functions.

To find the best hyperparameter combination, we perform a grid-search over all combinations of the three hyperparameters and perform 10-fold cross-validation for each combination. The combination with the highest overall macro precision, recall, and F1 scores is then re-trained on the entire training set (no development set) and scores on the testing set are reported. This process allows an unbias approach to finding hyperparameters that are good representatives of real-world population data.

6.5 End-to-end Evaluation

After each component of the three-part pipeline is evaluated individually, the final genealogical knowledge graphs are compared between the golden set of data and the model pipeline. This experiment assesses the system's ability to use all three components together and is based on comparing `RelGraphBuilder` objects, which contain final graphs featuring only coreference resolved entities that have a positive relation to some other entity in the text. Since there are no negative relations in these graphs, that score is not factored into the final calculations. Thus, the macro average scores in this case are only of positive relations.

To compare graphs of genealogical family trees, this thesis makes an algorithm that attempts to evaluate genealogical family trees that intends to mimic how a human would interpret and evaluate them. To start, every edge in the graph is converted to a tuple representing a unidirectional edge in the graph. The `Entity` at index 0 has a relation to the `Entity` at index 1. Now, it is sometimes the case where coreference resolved entities are correctly identified to have genealogical relations to other entities but not the entire entity has been resolved correctly or a mention has been used that captures only a portion of the resolved entities name. To still find these valid relations, a two-pass system is implemented.

On the first pass, all edges with entities that are fully correct and resolved and have the right direction, start indices, end indices, and so on are compared with each other, where true and false positives and negatives are calculated per label that those edges represent. Once those edges are removed from the evaluation stage, the second pass occurs.

On the second pass, the algorithm looks for `Entity` tuples that have strong confidence of being the same entities between the model's predictions and the golden data set. This means calculating a similarity score between each edge tuple in the model's predicted graphs and each edge tuple in the true graphs. Originally, a Levenshtein distance was used to compute similarities, but edit distances tend to favor strings that are very similar in lengths but can be vastly different names. Thus, a system of string subset comparisons and averages was used. For each entity in the tuple, the length of

$$Similarity((E_{1,1}, E_{1,2}), (E_{2,1}, E_{2,2})) =$$

$$(\frac{LCS(E_{1,1}, E_{2,1})}{min(len(E_{1,1}), len(E_{2,1}))} + \frac{LCS(E_{1,2}, E_{2,2})}{min(len(E_{1,2}), len(E_{2,2}))}) * \frac{1}{2}$$

Figure 6.5: Edge similarity formula

the highest common contiguous substring was found and divided by the total length of the smaller entity. The final similarity score of the tuple was the average of the values for both entities in a tuple where each was weighted by 0.5. If a pair of edge tuples was found to have a similarity score of > 0.80, then the edge is assumed to represent the same entities. Setting a high threshold at 0.80 was done to ensure that, for a pair of entities to be the same, both entities in both indices of the tuples must be over 80% confident to be similar or one entity must be 100% and the other at least more than 50% confident. This is done to mimic a human evaluator that would be able to see that even if some entities only have a portion of their true entity span, the entity represented is identifiable and different from other entities enough so that the identity can be deduced.

The formula is shown in Figure 6.5. LCS is an abbreviation for the longest common substring algorithm.

After true and false positives and negatives have been calculated per class on the same edges, macro averaged precision, recall, and F1 scores can be calculated over all relations across all documents.

Since the classifier model is of most importance in this thesis and the only model we train, an additional experiment is done that evaluates the end-to-end pipeline with gold labeled data for entities and anaphors in conjunction with the best trained classifier. This assesses the impact the model has on the end-to-end system and how much it detracts the system from reaching a theoretically perfect score. Furthermore, when comparing this value to the value of the prior classifier experiment, we can assess the efficacy of the model and how much other components of the pipeline contribute to the overall error.

6.6 Results

6.6.1 Named Entity Recognition

Scores for experiments concerning both finding named entities and their anaphors can be found in Table 6.2. This thesis chooses to look at “micro” scores as the most important metric for NER since the per-document detection is less important, and in this case the “micro” scores give good detail into how well the model performs in general over entity data. There is no class to skew the distribution, so the micro scores are acceptable.

SpaCy’s NER system has very high precision, though the recall is not so great. This is most likely due to the data set that it was trained on and its inability to find entity names in our data set that are not traditional English names. Data sets for training NER parsers often use older data sets that may include some inherent bias.

When combining named entities and their anaphors, the F1 scores of both models remain relatively similar. An interesting note is that entities and anaphors found with NeuralCoref have higher precision than with AllenNLP’s coreference resolution model, though drastically lower recall. Though precision is extremely important in an automated system for parsing genealogical texts, the plunge in recall would greatly affect future relation classification scores, so the AllenNLP + SpanBERT based coreference resolution model was chosen. The drop in precision likely comes from the fact that AllenNLP’s model likes to include additional span information for

	Macro (per-document)			Micro		
Model	P	R	F1	P	R	F1
SpaCy NER (no anaphors)	0.922	0.752	0.821	0.920	0.751	0.826
SpaCy NER + HF NeuralCoref	0.939	0.777	0.839	**0.937**	0.787	0.856
SpaCy NER + AllenNLP	0.906	0.852	0.870	0.894	**0.862**	**0.878**

Table 6.2: Precision, Recall, and F1 scores for Entity and Anaphor Evaluations

its anaphors that are extraneous. This thesis concludes that finding entities more entities (higher recall), albeit with extra fluff, would positively contribute to the pipeline.

6.6.2 Coreference Resolution

Isolating the coreference resolution scores, which can be found in Table 6.3, paints a much more clear picture of the two models. It really is no contest between NeuralCoref and AllenNLP's model. Similar to the NER evaluation, micro scores are preferred here to see how well the models perform in general entity tasks and there is no presence of a dominating class.

AllenNLP's coreference resolution model obtains higher precision than NeuralCoref and higher recall and F1 scores by a very wide margin. This is likely attributable to the use of state-of-the-art SpanBERT embeddings in contrast to NeuralCoref's naive embedding system. While these scores are good and much better than NeuralCoref's, we can observe that scores across the board for coreference resolution are generally pretty low. Coreference resolution is a notoriously hard problem, and in genealogical texts where anaphors can refer to multiple entities with ambiguous contexts, these spans of text may be difficult to resolve. This issue propagates itself into the final

	Macro (per-document)			Micro		
Model	P	R	F1	P	R	F1
HuggingFace NeuralCoref	0.706	0.521	0.587	0.728	0.508	0.598
AllenNLP + SpanBERT	0.814	0.705	0.739	**0.792**	**0.697**	**0.741**

Table 6.3: Precision, Recall, and F1 scores for Coreference Resolution Models

pipeline results in Table 6.6 in the "Parent-of" relation where this most commonly occurs.

Creating better coreference resolution models or performing fine-tuning on them for genealogical relation extraction could provide very beneficial boosts in scores for the future, as coreference resolution is an essential part of the pipeline for extracting **all** relationships in unstructured texts.

6.6.3 Fine-Tuned LUKE Classifier

For hyperparameter tuning of the model, the best results were found with the hyperparameter combination of `Epochs=4`, `Edge_chars=45`, and `Negative_rels=full`. Results per-label on the 10-fold averaged development set can be seen in Table 6.4. What this signifies to us about how the settings for this transformer is that with little data as in our case, the number of epochs for training has to be on the higher end of what was suggested by the authors of BERT (between 1 to 4 epochs) [13]. This, however, is a fraction of epochs typically needed for training deep neural networks from scratch, which can use hundreds of epochs for training.

Furthermore, having `Edge_chars=45` means that the model does benefit from having additional context surrounding spans of text for entity pairs, and that this context does sometimes include information necessary for classifying relations if it does not exist between the entities. Having too much context, `Edge_chars=60`, can obfuscate the relevant context for determining relationships.

Finally, and very importantly, is that undersampling generally had negative effects on model training, as seen by `Negative_rels=full`. This occurs likely for two reasons. First, as iterated before, by undersampling data we are artificially skewing the distribution the model learns that does not mimic real-world data. Because the development set was always unmodified and only the training set was undersampled, this can be confirmed. Furthermore, in all runs where data was undersampled, recall was higher but precision was always lower. Secondly, by omitting negative training examples, we might be omitting context that is important for the model to learn that is not typical of a relationship. In all, these hyperparameters still demonstrate that, given enough context, the ability to fine-tune pretrained transformers without additional pretraining (as is needed in the biomedical domain due to domain-specific entities) is a good use of the deeply learned and context-dependent word embeddings. Since these embeddings are learned from natural English language and genealogical texts are well within this domain, a final linear classifier layer and fine-tuning can achieve breakthrough results.

The results on the testing set are shown in Table 6.5. Again, these were only calculated after all parameters were chosen from the train/development sets.

6.6.4 End-to-end Evaluation

Finally, we have the results of evaluating the end-to-end system as is and with golden entities and anaphors. From Table 6.6 we can see that there is a relatively high precision across the board of relation labels. This signifies that, so long as the NER and coreference resolution systems find relevant entities, our model performs in-line with

Label	P	R	F1	Support
Child-of	0.934	0.904	0.919	156
No-rel	0.986	0.979	0.983	4317
Parent-of	0.782	0.758	0.770	128
Sibling-of	0.767	0.927	0.840	110
Spouse-of	0.744	0.877	0.805	106
Macro avg	**0.843**	**0.889**	**0.863**	**4817**
Micro avg	0.967	0.967	0.967	4817

Table 6.4: Precision, Recall, and F1 scores for best tuned classifier on DEVELOPMENT SET

Label	P	R	F1	Support
Child-of	0.951	0.817	0.879	71
No-rel	0.982	0.982	0.982	2141
Parent-of	0.785	0.805	0.795	77
Sibling-of	0.923	0.842	0.881	57
Spouse-of	0.762	0.889	0.821	72
Macro avg	**0.880**	**0.867**	**0.871**	**2418**
Micro avg	0.966	0.966	0.966	2418

Table 6.5: Precision, Recall, and F1 scores for best tuned classifier on TESTING SET

Adam and Jane married last summer. After relocating to a new city together and getting new jobs, they felt it the appropriate time to have a child, whom they named Marie.

Figure 6.6: An example of a tough-to-resolve Parent-of relation

results achieved from training the classifier on its own. Of course, having improperly resolved entities will impact this score. Just how much will be discussed shortly.

Another fact to note about Table 6.6 is that the recall for the “Parent-of” relation is significantly lower than recall for the other relations. This is because oftentimes parent-of relations are defined by an anaphor resolving to a couple and its relation to another entity. An example of this is in Figure 6.6, where the most relevant mention for the relationship to Marie, highlighted in pink, is “they,” highlighted in blue. This mention refers to Adam and Jane, highlighted in green. If this can be resolved by the coreference model, then this is potentially two relations to be created. The coreference resolution system, though, is relatively bad at finding these resolutions as they can often be ambiguous, and this propagates through the pipeline, impacting the recalls and dragging down the F1 score.

To answer the question of just how much each component propagates error to the end-to-end evaluation, we can compare results in Table 6.6 with results in Table 6.7, where a second experiment was performed fixing entities and anaphors to be golden. In this second experiment, precision and recall rates shoot up drastically as the only error propagating to the system comes from the classifier. Interestingly, these results perform better than results attained training the classifier alone (where entities are already given in the text spans). This shows us that there are cases where multiple relations are found between the same pairs of entities, but that using a confidence threshold can help choose only the most significant labels for entity pairs.

Label	P	R	F1	Support
Child-of	0.88	0.71	0.79	65
Parent-of	0.81	0.33	0.47	92
Sibling-of	0.77	0.72	0.74	46
Spouse-of	0.71	0.71	0.71	52
Macro avg	**0.794**	**0.616**	**0.676**	**255**

Table 6.6: Precision, Recall, and F1 scores for end-to-end model

Label	P	R	F1	Support
Child-of	0.95	0.83	0.89	65
Parent-of	0.95	0.87	0.91	92
Sibling-of	0.91	0.89	0.90	46
Spouse-of	0.78	0.90	0.84	52
Macro avg	**0.899**	**0.874**	**0.884**	**255**

Table 6.7: Precision, Recall, and F1 scores for end-to-end model with perfect entities and anaphors

To calculate the error contributed by the NER and coreference resolution models, we can subtract the macro averaged values in Table 6.7 from Table 6.6 because the classifier remains the same in both examples but the models are changed. Doing this, we find that the non-classifier models contribute a total of 0.105 precision, 0.258 recall, and 0.208 F1 score. While surely the classifier can be improved further, special attention must be given to the earlier two models in the system, which drag the overall performance lower.

Our experiments and results give us a completely transparent look into exactly what parts of the pipeline work well and which do not. Overall, our results prove our hypothesis of being able to effectively use transformers in the domain of genealogical relation extraction. With these results and the different errors found at each stage, we can discuss final remarks this thesis makes.

Chapter 7

CONCLUSION

To recap, this thesis sets out to bridge advancements made in the biomedical domain using state-of-the-art transformer models and applying them to genealogical relations rather than clinical ones. This was due to observing that prior works relied on imperfect, distantly supervised training data or noisier data augmentation techniques to obtain enough training samples for deep models. Observing this, this thesis proposed that by making use of transformer models massively pretrained on large corpora, we could attach a linear classification layer on top and perform fine-tuning with a small amount of gold-labeled data.

In doing so, this thesis showed that without feature-heavy approaches and a fraction of labeled data typically needed to train deep models, we could harness the inherently learned context-sensitive representations of words that BERT-based models, specifically LUKE, build to achieve great results on relation extraction. Specifically, LUKE's architecture further allows for reasoning about entities and their relations to each other, a key component missing from prior BERT-based works. Through rigorous 10-fold cross validation and hyperparameter tuning, macro averaged precision, recall, and F1 classifier scores of 0.880, 0.867, and 0.871, respectively, were achieved on the testing set. In conjunction with other improved models in this system, overall relation extraction scores of 0.794, 0.616, and 0.676, respectively, were obtained.

These scores represent a proof-of-concept that transformers might be the key to extracting genealogical relations from free-form text. Though this work is hard to compare to other genealogical works due to difference in relation labels, difference in types of documents, and so on, the results achieved are a clear indication of improvement over other models from prior work. With the contribution of a true golden data set and instructions on labeling and procedures for generating more data, more training instances can be crowd-sourced and used to improve scores further. Finally, a discussion and evaluation on individual components of the three-part document processing system suggest ripe areas of future work to be done that can drastically improve performance.

Following the state-of-the-art in other domains and improving the fine-tuning process for a LUKE-based model with more robust data could unlock great gains in making genealogical relationship data available to all.

Furthermore, all code and labeled data is publicly available in an online GitHub repository. The README file details how to use the data and how to properly view the code. The link for the repository is below:

https://github.com/ANGELLOPARR/luke-genealogical-thesis

7.1 Future Work

There are many moving parts in this system, which makes way for many potential improvements to be made.

7.1.1 Named Entity Recognition

It is possible that SpaCy's NER model is trained on a bias data set of names, creating a low recall for names that do not fit under the bias of its training data. It could be possible to combat this by looking into further training of this model on more diverse databases of names or creating them.

Furthermore, the NER model in this system could be replaced for a pretrained or fine-tuned BERT-based model. Since these models are breaking so many records on different NLP tasks, it is very reasonable that they could work for NER too.

7.1.2 Coreference Resolution

From the Results chapter, a clearly limiting factor in the end-to-end pipeline's results was the coreference resolution model. Improving the scores on this part of the pipeline, specifically in its recall score with resolving references to multiple entities, could serve to drastically improve results. Coreference resolution, however, is known for being a very hard problem that requires massive amounts of gold labeled data. A potential way to approach improving it, thus, could be taking the same approach this thesis took on our classification model: fine-tuning on domain-specific data.

As of right now, SpanBERT-based models hold the records for best coreference resolution scores on commonly used benchmarks such as the OntoNotes 5.0 dataset. While models utilizing SpanBERT embeddings, such as AllenNLP's coreference resolution model, can access and pretrain on this data, it could be worthwhile to hand-annotate a corpus with anaphors and their antecedents of interest with special attention to anaphors referring to multiple antecedents. Performing fine-tuning on a SpanBERT-based pretrained transformer with this new data can help "nudge" a transformer to learn better span representations for anaphors referring to multiple entities. If successful, this could improve recall scores and even precision, thereby boosting the performance of the entire system.

7.1.3 LUKE-based Classifier

While this thesis reasons that LUKE will be the best transformer to perform classification of relations between entities, a lot of the assumptions and hypotheses from this thesis could be challenged and evaluated to improve classification scores. For example, while undersampling was shown to worsen performance, oversampling could still serve to better performance, even if the data is noisy. There is a chance that, by using the right data augmentation approach, a classifier model can learn to interpret data through the noise and get better results.

Another improvement could be changing the way entity relation candidate pairs are made to include some confidence score before classification for if entities can be related. In doing this, the amount of negative relations in a document can be reduced and make way for a more balanced distribution of relations to help a model learn positive relations more efficiently. A similar tactic could be making use of predicted relations "on-the-fly" to, once prior entity relations are classified, disqualify entity pairs further in the document based on relations from prior entities.

Furthermore, while gold-labeled data is used and is highly helpful for learning, using distantly supervised approaches with a knowledge base to create silver-labeled data might bolster performance despite the added noise in positive relations.

BIBLIOGRAPHY

[1] AllenNLP demo. `https://demo.allennlp.org/coreference-resolution`.

[2] The ely ancestry : lineage of richard ely of plymouth, england, who came to boston, mass., about 1655 | ancestry®. `https://www.ancestry.com/search/collections/10261/`.

[3] Index to the register of marriages and baptisms in the parish of kilbarchan, 1649-1772. `https://www.forgottenbooks.com/en/books/IndextotheRegisterofMarriagesandBaptismsintheParishofKilbarchan16491772_10749172`.

[4] Label studio – open source data labeling. `https://labelstud.io/`.

[5] Miller funeral home records, 1917-1950, greenville, ohio. `https://www.familysearch.org/terms`.

[6] NeuralCoref 4.0: Coreference resolution in spaCy with neural networks. . original-date: 2017-07-03T13:04:16Z.

[7] spaCy models - 2.1.0. `https://github.com/explosion/spacy-models/blob/477f852c534c686fe78717a33a5a77e4268a27cf/meta/en_core_web_sm-2.1.0.json`. original-date: 2017-03-14T11:15:20Z.

[8] D. Bahdanau, T. Bosc, S. Jastrzębski, E. Grefenstette, P. Vincent, and Y. Bengio. Learning to compute word embeddings on the fly. type: article.

[9] L. Baldini Soares, N. FitzGerald, J. Ling, and T. Kwiatkowski. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the*

57th Annual Meeting of the Association for Computational Linguistics, pages 2895–2905. Association for Computational Linguistics.

[10] J. Beidler. FamilySearch using artificial intelligence to help index. `https://www.roots-branches.com/familysearch-using-artificial-intelligence-to-help-index/`.

[11] T. S.-T. Chu. Genealogy extraction and tree generation from free form text.

[12] 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 Human Language Technology Conference of the NAACL, Main Conference*, pages 296–303. Association for Computational Linguistics.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding.

[14] J. Efremova, A. M. García, and J. Zhang. Towards population reconstruction: extraction of family relationships from historical documents. page 9.

[15] 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 - CIKM '98*, pages 52–59. ACM Press.

[16] M. Grandini, E. Bagli, and G. Visani. Metrics for multi-class classification: an overview. type: article.

[17] A. Harnoune, M. Rhanoui, M. Mikram, S. Yousfi, Z. Elkaimbillah, and B. El Asri. BERT based clinical knowledge extraction for biomedical knowledge graph construction and analysis | elsevier enhanced reader.

[18] K. He, L. Yao, J. Zhang, Y. Li, and C. Li. Construction of genealogical knowledge graphs from obituaries: Multitask neural network extraction system. 23(8):e25670. Company: Journal of Medical Internet Research Distributor: Journal of Medical Internet Research Institution: Journal of Medical Internet Research Label: Journal of Medical Internet Research Publisher: JMIR Publications Inc., Toronto, Canada.

[19] S. Hochreiter and J. Schmidhuber. Long short-term memory. 9(8):1735. Publisher: MIT Press.

[20] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. SpanBERT: Improving pre-training by representing and predicting spans. type: article.

[21] K. Lee, L. He, and L. Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference.

[22] C. Lin, T. Miller, D. Dligach, S. Bethard, and G. Savova. A BERT-based universal model for both within- and cross-sentence clinical temporal relation extraction. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 65–71. Association for Computational Linguistics.

[23] P. Lindes, D. Lonsdale, and D. Embley. Ontology-based information extraction with a cognitive agent. 29(1). Number: 1.

[24] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. type: article.

[25] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. type: article.

[26] A. Makazhanov, D. Barbosa, and G. Kondrak. Extracting family relationship networks from novels. type: article.

[27] C. Manning. Coreference resolution.

[28] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press.

[29] G. Nagy. Near-perfect relation extraction from family books. In J. Lladós, D. Lopresti, and S. Uchida, editors, *Document Analysis and Recognition – ICDAR 2021*, Lecture Notes in Computer Science, pages 477–491. Springer International Publishing.

[30] M. Perrow and D. Barber. *Probabilistic Tagging of Unstructured Genealogical Records*. IDIAP.

[31] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. type: article.

[32] B. Ranjbar-Sahraei, H. Rahmani, G. Weiss, and K. Tuyls. Distant supervision of relation extraction in sparse data. 23(5):1145–1166. Publisher: IOS Press.

[33] D. Santos, N. Mamede, and J. Baptista. Extraction of family relations between entities. page 12.

[34] I. Q. Sayed. Issues in anaphora resolution. page 20.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need.

[36] R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini, M. El-Bachouti, R. Belvin, and A. Houston. OntoNotes release 5.0. Artwork Size: 2806280 KB Pages: 2806280 KB Type: dataset.

[37] T. Wolf. State-of-the-art neural coreference resolution for chatbots.

[38] C. J. Woodbury. Automatic extraction from and reasoning about genealogical records: A prototype. page 60.

[39] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto. LUKE: Deep contextualized entity representations with entity-aware self-attention.

[40] X. Yang, Z. Yu, Y. Guo, J. Bian, and Y. Wu. Clinical relation extraction using transformer-based models. type: article.

[41] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45. Association for Computational Linguistics.