Deciphering the U.S.Diplomatic Documents with NLP and Graph Data Science

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Deciphering the U.S. Diplomatic Documents with NLP and Graph Data Science

Master's Thesis

Gökberk Özsoy
goezsoy@ethz.ch

Swiss Data Science Center
ETH Zürich

Supervisors:
Dr. Luis Salamanca
Prof. Dr. Fernando Perez-Cruz

June 4, 2023

Abstract

Any corpus of historical records is beyond a simple collection of documents, as it implicitly hosts a vivid network of connections between entities mentioned within. In this thesis, we introduce a paradigm for processing large archives as a whole, which enables to uncover all their value. Main pillar of this paradigm is parsing and storing the corpus as a knowledge graph. This data representation is not only flexible, light-weight and exhaustive, but also reachable by a broad community from domain experts to data-driven scientists. In response, it will nurture a new set of domain-specific questions, which will be answered by computational approaches powered by natural language processing, and graph algorithms. We apply our methodology to the FRUS corpus (records of U.S. diplomatic cables) to demonstrate how it facilitates the understanding of complex fields of diplomacy, foreign relations, and politics. As a result, an extensive knowledge graph is built, enabling explorations unattainable using solely the text transcripts, which further boosts the value of the corpus. Throughout this thesis, we provide broad analysis on this knowledge graph to prove the potential of our paradigm.

Contents

CHAPTER 1

Introduction

The Foreign Relations of United States (FRUS) [1] series is a historical record collection that reflects major U.S. foreign policy decisions and diplomatic activities from 1861 to 1988. It is produced by the Department of State's Office of the Historian, and consists of 543 volumes over 127 years. Each of these volumes contain documents from presidential libraries, Departments of State and Defense, National Security Council, Central Intelligence Agency, Agency for International Development, and foreign affairs agencies.

The publishing principle states that documents that are selected for declassification must totally cover the policy targets of the U.S. in that period of history. Hence, there will be no alteration, deletion, or concealment in the case of defect or weakness. In this way, FRUS can be seen as an accurate, historically correct, reliable, and complete entering point to the recent world history, but from a U.S. point of view.

Many of the volumes include person and term (e.g. abbreviations of institutions and political terms) annotations. Each annotated person or term is complemented with a description at the beginning of the volume. In addition, most of the documents have sender, receiver, and city metadata. Thus, FRUS corpus is far beyond than a monotonous concatenation of documents.

Knowledge graphs (KGs), also known as semantic networks, have emerged as an abstraction for representing real-world entities and relationships between them in a machine-readable format.

A KG is a directed labeled graph which has four main components: *node* (entity), *edge* (relationship), *attribute* (property), and *label*. A node can be anything (i.e. objects, concepts, or events), an edge describes a connection between a source node and a target node with a particular direction. It must have a type to classify what kind of relation exists between the two. An attribute further stores information about a node or an edge. Labels group nodes into sets where all nodes in a set share a common identity. We embody these concepts in the example below (in Neo4j Cypher notation [2]):

```
(p1:Person {name: 'Richard Nixon', born: 1913})
(p2:Person {name: 'Jimmy Carter', born: 1924})
(r:Role {name: 'president of the U.S.A'})
(p1)-[:WORKED_AS {started:1969, ended:1974}]->(r)
(p2)-[:WORKED_AS {started:1977, ended:1981}]->(r)
```

where `p1`, `p2`, `r` are nodes, `WORKED_AS` is edge, `Person`, `Role` are labels, and `name`, `born`, `started`, `ended` are attributes.

KGs usually integrate several sources that host data in different formats. Through domain expertise and specific needs, a schema (abstract construction plan) is defined, and extractions of entities and relationships from different sources are outlined accordingly. To give an example, in this work, we first decide a schema with a focus on political science needs, then integrate two sources, FRUS corpus and Wikidata [3], into one KG, after some steps of processing suitable to the schema.

Wikidata is one of the general-purpose open-world KGs. Notable others are as DBpedia [4], Freebase [5], and Google KG. Among all, Wikidata acts as a structured version of the generic knowledge served in Wikipedia. It focuses on *items* which represent any kind of topic, concept or object. Each item has a label, a description and a number of aliases. Items are uniquely identified with a `Q` followed by a number. *Statements* hold various information about an item (subject) and consist of a *property* and a *value* (object item). Properties are identified with a `P` followed by a number. Let us give an example:

```
Richard Nixon (Q9588) is an item.
Jimmy Carter (Q23685) is an item.
President of the United States (Q11696) is an item.
position held (P39) is a property.
Q9588 - P39 - Q11696 is a statement, where Q11696 is the value.
Q23685 - P39 - Q11696 is another statement.
```

Apart from general-purpose KGs, using KGs in specific domains, such as retail, entertainment, healthcare, finance, politics, and education, is of great importance to better capture the complexity of these domains, henceforth enabling a more complete and insightful investigation [6][7].

FRUS corpus is invaluable to historians, but dealing with such a large corpus end-to-end is beyond human capability. Furthermore, it does not include generic information about an annotated person's gender, education, citizenship, role, and occupation, or a country's region and cities. Assuming we would have this extra piece of information, it is still hard to integrate all these seamlessly for an historian or a political scientist to benefit from.

Here, we advocate that KGs can facilitate the understanding of complex relations between different actors in FRUS corpus. For this, we propose a pipeline that is generic enough to be applied to other text corpora in different fields. For constructing this pipeline, we do parsing and enrichment. Former includes parsing documents, and doing the required data preprocessing to ensure the consistency. Here, parsing returns document's metadata such as sender, receiver, city and text. Latter may include NLP methods that helps extracting additional entities from texts. In our case, these have been topic modeling, named entity recognition, redaction extraction, and sentiment analysis.

The result is a comprehensive and consistent KG that will allow historians to reach information easily, formulate their hypothesis, and provide necessary answers. One can use it at different levels: from simple exploration to advanced graph machine learning methods. For us, potential research directions include how political actors change their political positions, how the perception of the U.S. towards specific topics shifts over time, and which countries, roles, persons, topics or institutions were popular in certain periods, among others. In this work, we showcase these and other interesting applications possible with our KG.

In summary, our main contributions are:

- To the best of our knowledge, we propose the first domain-specific KG of U.S. diplomatic relations, integrated with a text corpus, that enables dynamic analysis of entities for a substantial 127 years.
- We present a paradigm to encode similar corpus or archival records in the form of a KG, whenever entity-relation notions might be of interest for further capturing the contained phenomena. This enables to easily harness NLP methods to further process the related documents, and complement the graph.
- We adapted graph algorithms for obtaining dynamic entity embeddings, estimating role and person importance scores, and finding missing relationships in the KG.
- We analyze several important events in the world history from a data science perspective using our KG.

The rest of this report is structured as follows: Section 2 provides a literature review on NLP, KGs, and political science. Section 3 explains in detail how we constructed the KG, and the idea behind each graph algorithm for analyzing the FRUS corpus. Section 4 evaluates parts of the pipeline qualitatively and quantitatively. Section 5 discusses the results from the point of view of their value for an historical analysis of the data. Section 6 concludes with a discussion of the method, its limitations, future avenues and possibilities.[1]

[1]Source code is available at `https://github.com/gozsoy/decipher-frus`.

CHAPTER 2

Previous Work

2.1 Natural Language Processing

Natural language processing (NLP) is an interdisciplinary subfield of linguistics and computer science that develops algorithms which can understand, interpret and generate human language.

To understand the field's current progress, it is enough to observe how quickly human baselines were surpassed by NLP models in rigorous benchmarks, such as SuperGLUE [8]. It includes tasks such as word sense disambiguation, co-reference resolution, natural language inference, and question answering.

In this thesis, we will use a diverse set of NLP tools, from basic string algorithms to complex Transformer-based architectures, for extracting additional valuable information, and interpreting the historical facts and events reported in FRUS corpus.

String Algorithms. String distance metrics measure differences between two string sequences. Levenshtein distance is the minimum number of single character edits (i.e. insertion, deletion, substitution) necessary to change one string to other. Damerau-Levensthein distance includes all edit operations of Levenshtein distance, plus it allows transposition (i.e. swap) of two adjacent characters. If s_1 = *cs eth*, and s_2 = *c teh*, Levenshtein distance becomes 3 (delete, delete, insert), while Damerau-Levensthein distance becomes 2 (delete, swap).

Jaro similarity is another string distance metric, computed as following:

$$sim_j = \begin{cases} 0, & \text{m=0} \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right), & \text{otherwise} \end{cases}$$

where m is the number of matching characters, t is number of transpositions, and $|s_i|$ is length of string s_i. Here, two characters from s_1 and s_2 are considered matching if they are the same, and not far apart from each other $\lfloor \frac{max(|s_1|,|s_2|)}{2} \rfloor - 1$ characters. Jaro-Winkler similarity favors strings in the magnitude of the length they match from the beginning.

$$sim_w = sim_j + lp(1 - sim_j)$$

where sim_j is Jaro similarity, l is length of the common prefix up to max of 4 characters, and p is scaling. If $s_1 = cs\ eth$, and $s_2 = c\ teh$, Jaro similarity is 0.87, and Jaro-Winkler similarity is 0.89, if p is 0.1.

Another set of algorithms relevant to this thesis are methods to compute textual richness (a.k.a. lexical richness, vocabulary diversity) [9]. They measure the range and diversity of the vocabulary deployed in a string.

$$Type\ Token\ Ratio = \frac{t}{w}, \quad Corrected\ Type\ Token\ Ratio = \frac{t}{\sqrt{2w}}$$

where t is unique token count, w is total token count. Latter is introduced to mitigate effects of very long texts.

Named Entity Recognition. Named Entity Recognition (NER) deals with labeling sequences of words in a text which are the names of persons, organizations, countries, nationalities, companies, geographical locations, events, dates, monetary values, measurements, numerals etc. Reformulated as a machine learning task, conditional random fields [10], and more recently BERT-based methods [11] reach state-of-the-art in datasets such as CoNLL 2003 [12]. NER is especially useful for our work since person and institution annotations in some volumes are missing in FRUS corpus, hence it could be used to extract these further for a more complete KG.

Topic Modeling. Topic modeling is commonly used for automatically exploring latent topics and themes within a text corpus. Topic modeling algorithms are unsupervised, and the resulting topics are actually groups of words, which need further interpretation by humans (i.e. naming a topic, or reducing the topic count).

Latent Dirichlet Allocation (LDA) [13] is a generative model that represents documents as mixtures of topics, and topics as mixtures of words. It processes each document in a bag-of-words format, ignoring the word order, and learns the LDA parameters (per document topics distributions, and per topic word distributions) by word co-occurrence statistics. Optimization is done via Bayesian inference. A major drawback of LDA is that it cannot capture word and document semantics. Dynamic LDA [14] captures the evolution of topics in a temporally organized text corpus (i.e. documents with dates) with the following modification: In LDA, the parameters of both Dirichlet distributions which are used as priors for choosing topic proportions per document, and word proportions per topic are single, hence static. In Dynamic LDA, they are modelled as random variables depending on their previous values. This enables understanding how topics may gain or lose importance, and how these topics can be characterized by different words over time.

A semantically more powerful model is BERTopic [15], which leverages Section 2.1 Sentence Transformers for topic identification. It starts by obtaining document embeddings using S-BERT, then aggregates these with a density-based clustering algorithm (e.g. HDBSCAN). Finally, it concatenates all documents for each cluster to create one big document and computing TF-IDF scores to identify most critical and unique words for each cluster. The resulting topics are more interpretable than LDA because of the semantical embedding process. However, it is not as mathematically grounded as LDA. A dynamic version exists as well with added components to above pipeline.

Sentiment Analysis. Sentiment analysis identifies the emotional tone in text. It is already a huge field in NLP, with lots of different variants and applications. Specifically for this project, we focus on target based sentiment analysis, which aims to identify a fine-grained sentiment towards a specific entity in a given text. For example, in "The pancakes were delicious, but service was slow.", it should find a positive sentiment towards pancakes and negative towards service. In Sentihood dataset [16], each location (urban neighborhood) mentioned in a text has sentiment labels. Authors treat the problem as sentence pair classification task with BERT [17]. In addition to the original sentence, they use an auxiliary sentence appended to it such as *What do you think of <target>?* as input to BERT, where the label is <target>'s label from dataset. This solution is only partial, and does not fit to most of the real-world corpora, including ours.

Emotionmeter [18] finds a text's emotionality score. First, the authors train word embeddings on a political corpus. Then, using a list of affection and cognition words, they compute similarity of a given text's average word embedding to that of affection and cognition words. Finally, emotionality is defined as the ratio of its affection score to cognition score. Even though using average word vectors for such a delicate task might not capture subtleties, it is a noteworthy, creative and unsupervised perspective to sentiment analysis.

Sentence Transformers. A Transformer [19] is a revolutionary deep learning model which introduced self-attention mechanism and parallel token processing, allowing efficient handling of long-range dependencies in text. BERT [20] is a Transformer based model that is pre-trained on two tasks: predicting missing words in a sentence, and predicting if the second sentence is the continuation of the first. This way, it learns to capture contextual information of text, and excels in tasks such as text classification, and question answering.

Sentence Transformers (S-BERT) [21] is the product of a supervised contrastive learning schema that aims to map semantically similar sentences closer in the embedding space. Given a text pair, both texts are passed through a BERT independently, then cosine similarity is computed between the resulting embeddings where the label is -1 for dissimilar pair, and 1 for similar. This type of BERT is dubbed as bi-encoder, in contrast to cross-encoder as in classical BERT setting where it encodes both sequences in one pass. In addition, it is siamese because the model weights are the same when encoding two different inputs. S-BERT produces more meaningful embeddings, at the sentence level, than BERT, thanks to its training regime, and is useful for semantic textual similarity, semantic search, paraphrase mining, information retrieval, and clustering.

2.2 Knowledge Graphs

This section presents several highly active research areas [22] relevant to this thesis in the intersection of KGs and NLP.

2.2.1 Entity & Relation Extraction

An important step of constructing KGs is extracting semantic relationships between entities from raw text. In earlier attempts, the task is solved via a two-stage pipeline, where the first stage is named entity recognition, and the second is relation type classification between recognized entities [23].

An end-to-end solution, REBEL [24] frames the problem as seq2seq task hence uses Encoder-Decoder Transformer architecture, where the input is a raw sentence, and the outputs are triplets present in that sentence. They are generated in the same order as in the sentence, and special tokens are used for annotation of head, tail entities, and their corresponding relation for each triplet. They use a dataset compiled from Wikipedia abstracts with the most common 200 relation types retained, which is more diverse than other available datasets.

In FRUS corpus, document metadata as well as in-text annotations naturally creates relations by the file structure. On the other hand, semantic relations (the focus of this section) is harder to obtain. FRUS corpus is abstract, and complex. Information is present mostly in dialogue format, and full of hidden mentions to historical facts and events. Let alone if 200 relation types from REBEL would be enough to map historical phenomena, it may not even be possible to define what has discussed in the triplet format (e.g. See the sentence *Vietnamese people have faith in their rulers.* Which are the entities, and relations?).

2.2.2 Entity Linking

Entity linking (disambiguation) is the task of matching each entity mention in a given input text with the entries available in a knowledge base. It is called

Wikification if the knowledge base is Wikidata [3].

In BLINK [25], authors devise a scalable zero-shot two-stage pipeline. At first, they use two pretrained BERTs, one for embedding Wikipedia entry descriptions, and the other for embedding the input sentence. The dot products between sentence's CLS token and those of Wikipedia entries are computed, and top ones are regarded as candidates. In the second stage, the sentence and each candidate is concatenated, and fed into a pretrained BERT for cross-encoding. The maximum score candidate is then the corresponding Wikipedia entry.

ExtEnD [26] reformulates the problem as span extraction task similar to question answering using BERT. Input sentence containing entity mention is concatenated with titles of Wikipedia entries, and is fed to a Transformer. The output is the start and end tokens which indicates predicted Wikipedia entry.

2.2.3 Link Prediction

Link prediction (a.k.a Knowledge Graph Augmentation) is the task of predicting missing or potential relations between entities in a KG. Even state-of-the-art KGs suffer from incompleteness. For instance, it has been reported that over 70% of person entities have no known place of birth, and over 99% have no known ethnicity in FreeBase [27].

A common approach to this problem is obtaining entity embeddings using one of the embedding techniques, some of which are mentioned in Section 2.2.4. Later, an entity pair is featurized using an aggregation function such as Hadamard product, or L2-distance between pair's embeddings. The same entity pair is labeled as adjacent or not depending on whether they are connected or not in the KG. A binary classifier learns how to map connectivity patterns embedded in features to labels. Finally, the trained model can be used to predict new relationships between entities that are not already connected.

2.2.4 Knowledge Graph Embeddings

KG embeddings are used to create dense vector representations of entities (and sometimes relations) to be used in downstream tasks. These embeddings give information about entities' similarities and network structure.

Node2vec [28]. Node2vec for graphs is simply what Word2vec is for text. The notion of co-occurrence in a window is done by random walks. Given a starting node, we select a neighbor of it randomly, and move to this node, then repeat the same process for a pre-defined number of steps. A combination of breadth-first and depth-first search provides local and global perspective from graph structure, respectively. The resulting node chains (i.e. sentences) is used to train embeddings which maximizes their likelihood, combined with negative sampling. This algorithm works in homogeneous graphs, and does not produce edge embeddings.

TransE [29]. TransE is an algorithm for learning both entity and relation embeddings from multi-relational graphs. Given a triplet (h, l, t) where h is source entity, t is target entity and l is relation between them, it learns embeddings to ensure h + l $\approx$ t. The training objective is triplet loss where the euclidean distance d(h+l, t) is minimized against the distance of a corrupted non-existing triplet.

FastRP [30]. FastRP (Fast Random Projections), claims to be 4000 times faster than Node2vec, by using very sparse random projections. Starting with random node embeddings, it iteratively averages over neighboring nodes. At the end, a node's embedding is decided by a radius of neighborhood depending on the iteration count. This optimization free method makes it scalable. Like Node2vec, it does not produce edge embeddings.

Suitable inputs to machine learning models, these embeddings are widely used in different fields, from fake news detection [31] to collaborative filtering [32].

2.2.5 Centrality Algorithms

Centrality algorithms find a particular entity's importance (in terms of numerical score) in the KG. Each version defines its own perception of importance, hence suitable for different applications.

Degree centrality is the count of incoming or outgoing (or both) edges. *Closeness centrality* is the inverse of farness, where farness is the sum of distances from a particular node to all other nodes. Distance is computed as shortest path between two nodes. *Betweenness centrality* is how many times a node occurs on the shortest path between two other nodes. *Eigenvector centrality* computes the score of a node as a function of scores of its neighbors. Hence, a high score means the node itself is connected with many important nodes. The scores are initialized to a default value and they are computed iteratively until convergence. *PageRank centrality* [33] is a variant of eigenvector centrality.

2.3 Political Science

The graph-based FRUS corpus presented in the current thesis will facilitate research in political science. In this field, scientists commonly use off-the-shelf NLP methods to analyze text, aiming at capturing political patterns. However, KG use is rather rare, but pioneering works have proved its potential. In this section, we will explore these two directions in more detail.

2.3.1 Political Science & NLP

U.S. Congressional Record spans from 1858 to 1994 and includes 6 million speeches uttered by the congress members. It is a huge dataset, and authors use it to analyze emotionality change over time depending on political party, gender, era, etc. [18] using their emotionmeter explained in Section 2.1 Sentiment Analysis.

State Department records from years 1973-1979 transmitted to and from Iran is used to conclude that officials in Iran had actually reported on the protests leading to Islamic Revolution, but D.C. officials could not comprehend the severity of the situation. The authors use basic traffic statistics, and dictionary based sentiment analysis to obtain the reported results [34].

UK Government Web Archive [35] holds UK central government information published on web from 1996 to the present day. Researchers develop a method for efficient semantic search in this archive [36]. They first extract named entities within each text, then train document embeddings via doc2vec [37] which enables searching on entities, or fetching similar documents for a topic.

In an another work in the same direction, document embeddings is used to capture dynamic shift of political actors through time [38]. In their formulation, all documents produced by a party for a specific time bin are concatenated into one, so the document embedding can be treated as political party embedding. In this way, they could compare ideology shifts reflected by party embeddings, and also by comparing to keywords such as abortion, or immigration. They use congressional records for US, UK, and Canada.

Lastly, a Greek parliamentary corpus is presented which spans from 1989 to 2020 which consists of 1 million speeches with speaker metadata. The authors use this corpus for analyzing word use change in Greek language though it is a short span and a specific domain [39].

2.3.2 Political Knowledge Graphs

An early effort for political KGs is BBC Politics Ontology [40]. It is an ontology particularly designed for local government and elections with UK Local, and European Elections in May 2014.

POLARE [41] concerns with producing a KG about political agents in Brazil. A major data source is House of Deputies and Senate, but crowd sourced contributions have been made, although laborious and not scalable. They focus on persons, organizations and their connections, but also capture legislative aspects and electoral processes.

On another work, authors compile a fusion KG obtained from heterogeneous sources including official sources about Australian Parliament members, Australian Twitter's users tweeting on politics, and related metadata, WordNet, and Politics domain ontology [42]. They use the final KG for producing KG embeddings to show persons with similar political ideologies place closer in the embedding space.

Moreover, tailor made task specific political KGs exist as well. For example, authors extract belligerents table from each info box of Wikipedia articles on conflict. Then, they create a friend-enemy graph, obtain node embeddings and try to classify if a particular pair of entity is enemy or friend [43].

Political perspective detection is the task of obtaining political leaning of a text piece (e.g. leftist, rightist). In this direction, authors craft a domain-specific political KG to be used as an external source base [44]. For this, they select a set of political actors from past decade U.S. politics, then use their Wikipedia pages and simple relations (homeland, party, etc.) as base graph. They support it with expert views from two think-tanks, that provide information about which political actor supports/opposes others. This input is precious, and highly useful as it provides hidden patterns beyond facts, yet it is not scalable especially for a corpus like ours.

At this point, our difference is that we have a rich and interesting set of intelligence documents on a long span. This allows us to analyze how entities in our KG change their positions over time. In other words, our KG is dynamic in contrast to first 3 works being static. In addition, we aim our paradigm to be fully automatic with minimal manual work, general, extensive, and sourced by public datasets, in contrast to last 2 works.

Chapter 3

Methodology

This thesis has two main parts: KG construction, and its analysis using both basic KG probing and more advanced graph algorithms. Former describes the conversion of FRUS corpus into a structured knowledge base supported by Wikidata, and latter shows how to gain insights about historical actors automatically and in scale using constructed KG. Sections and subsections are matched with corresponding files in codebase, which is further explained in Section 4.3.

3.1 Knowledge Graph Construction

In this section, we start by exploring the XML structure of FRUS files, and how to parse it appropriately. We then explain our algorithms for person and term unification and wikification. In the second part, we elaborate KG enrichment, which consists of named entity recognition, redaction extraction, topic modeling, and target based sentiment analysis.

3.1.1 Parsing & Base Graph Population

XML Structure Discovery

FRUS corpus is publicly available in digital format[1]. Each of its XML files represents a single volume, and they have been prepared according to Text Encoding Initiative (TEI) P5 guidelines. The developers also explain project-specific encoding guidelines and conformance requirements in a schema[2]. In FRUS, volumes are simply the concatenation of all documents on a particular topic for a certain president's term. A document is the declassified historical record itself.

At this point, we need to understand this schema, parse it appropriately, and estimate importance of XML tags and attributes on the scale if they can add knowledge to our graph. XML is an inherently hierarchical data format, and the

[1]https://github.com/HistoryAtState/frus

[2]https://github.com/HistoryAtState/frus/blob/master/schema/frus.odd

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI xml:id="frus1969-76v30"
    <teiHeader>...
    </teiHeader>
    <text>
        <front>...
        </front>
        <body>
            <div type="compilation" xml:id="comp1">
                <div type="document" xml:id="d1">
                    document metadata, title, free text
                </div>
                other documents...
            </div>
        </body>
        <back>...
        </back>
    </text>
</TEI>
```

Figure 3.1: XML hierarchy of FRUS volumes

best way to represent it as a tree. For parsing, we use Python's standard library ElementTree XML API[3]. We visualize the common hierarchy of files in FRUS in Figure 3.1. Note that small variations occur from this structure across files. Please refer to one of the volumes for further insight.

`<teiHeader>` has `<fileDesc>`, `<encodingDesc>`, `<revisionDesc>` children tags, which hold metadata information about the current volume, paper to digital layouts and style definitions, and revision history, respectively. Metadata information include series name, sub-series name, volume number, volume name, publication information.

`<front>` has `<preface>`, `<table-of-contents>`, `<sources>`, `<terms>`, and `<persons>` children tags. Among them, `<terms>` and `<persons>` have particular importance, because they hold annotation id, name, and description information. Persons and terms in `<div type="document">` are tagged with these ids, which we can easily parse and extract. An item from `<persons>` is depicted in Figure 3.2.

`<body>` holds the documents, which are our focus. From now on, we refer to `<div type="document">` as document tag. For document tag's subtype attribute, FRUS has two options: 'historical-document' or 'editorial-note'. Latter

[3]https://docs.python.org/3/library/xml.etree.elementtree.html

```
<item>
<hi rend="strong">
<persName xml:id="p_AC8">Albert, Carl</persName>,</hi>
Democratic Congressman from Oklohama
</item>
```

Figure 3.2: An exemplary person annotation in FRUS

is not historical, but editor's notes explaining particular events in the course of a particular era. Document tag's children tag variety (42 different tags) is the highest across all XML structure. We investigated each tag, and its contents to understand which stores important information about the document. `<list>`, `<p>` hold free text, `<head>` holds the title of document, chapter or compilation, `<persName>`, `<gloss>` hold person and term ids, respectively, whose annotation information is given in `<front>`. `<placeName>` is the city the document was sent from. `<signed>` holds the signed person's name. These are the tags we will use in KG construction. The tags we do not utilize in this thesis but might be useful in future iterations are: `<note>` holds footnotes, `<ref>` cross references to other documents, `<table>`, `<row>`, `<cell>` represent table occurring in free text. The following are the tags which do not hold useful information, and mostly about style and layout: `<pb>`, `<hi>`, `<seg>`, `<idno>`, `<lb>`, `<pageline>`, `<attachment>`, `<opener>`, `<date>` (already exists as attribute in `<div type="document">` tag), `<label>`, `<item>`, `<quote>`, `<salute>`, `<affiliation>`, `<postscript>`, `<closer>`.

Basic Extraction

Given the exploratory nature of the analysis currently intended of the FRUS volumes, and with the light of an abstract construction plan in our mind, we parse the files. For each file, we iterate over its document tags and extract the following fields: *document id*, *document subtype* (i.e. historical-document or editorial-note), *date*, *year*, *presidential era*, *title*, *source* (i.e. where the document is stored physically), *persons who sent the document*, *persons who received the document*, *persons mentioned in the document*, *terms mentioned in the document*, *city where the document was sent from*, *institutions who sent the document*, *institutions who received the document*, *free text* (i.e. the body of the document).

1. *Document id* is the concatenation of volume number and `xml:id` attribute of document tag, such as `frus1969-76v30_d1`.
2. *Document subtype* is subtype attribute of document tag.
3. *Date* and *year* are obtained from `doc-dateTime-max` attribute of document tag.

4. *Era* is obtained automatically by comparing the document's date with presidential term dates. Presidential term dates are obtained from Wikipedia[4].

5. *Title* is obtained from extracting text residing inside the `<head>` child of document tag.

6. *Source* is obtained from the first `<note>` child of document tag which has source attribute. This is in accordance with FRUS guidelines.

7. *City* is obtained by finding `<placeName>` child, if exists, and extracting text inside of it.

8. `<persName>` child of document tag has the the following attributes: `corresp` with values of person identifiers, `type` with values of `from` or `to`. Persons who sent the document are annotated with `type="from"`, and persons who received the document are annotated with `type="to"`. Persons mentioned within the document do not have `type` attribute. All annotations have `corresp` attribute.

9. `<gloss>` child of document tag has `target` attribute with values of `from`, `to`, or term ids. Here, `from` and `to` hold free text indicating the document's sender and receiver institution, such as Embassy in Ankara or White House. When `target`'s value is term id, it refers to term annotation at that point. Thus, `<gloss>` has dual duty.

10. Only `<p>`, and `<list>` children of document tag contain free text. If we extract these from footnotes (i.e. `<note>`) and concatenate their contents, we obtain the document's free text.

All these efforts lead us to the KG structure shown in box **A** of Figure 3.3.

Person Unification

Unification, record linkage, or entity resolution, is the process of finding same real-world entity with different names across volumes. For example, the American president Richard Nixon is referred to as "Richard M. Nixon", "Richard Milhouse Nixon", "Richard Nixon", and "Nixon Richard" throughout FRUS. This is without considering possible typos that might exist in the text. Besides, as FRUS editors confirm, identifiers are useful only for single-volume purposes, so we cannot understand automatically that these names refer to same person. To solve this issue, we implement an iterative algorithm for unification:

1. We extract persons from every volume which has `<persons>` tag within `<front>`. Each person has a name, volume-specific identifier (id), and a description (mostly about that person's role in that volume or in general).

[4] https://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States

2. We reduce exactly matched names (i.e. all exact "Richard Nixon"s) and concatenate their ids and descriptions.

3. We reduce names with exactly same words but different combinations (i.e. "Richard Nixon" and "Nixon Richard") and concatenate their ids and descriptions. We do this by splitting each name by whitespace, and concatenating its words according to alphabetical order. We call this order *unified name*. For example, "Richard Nixon"s unified name is "Nixon Richard". This and previous steps cannot generate false positives, because they are deterministic.

4. The last step considers reducing near-duplicate names and obvious misspellings. We define misspelling between a name pair if its Damerau-Levensthein distance is at most 1. We define a name pair near-duplicate if it has at least two common words, plus either Damerau-Levensthein distance is at most 5 or Jaro similarity is at least 0.9. These distance conditions are subjective, and the cutoff values could be treated as hyperparameters. We think these values are conservative to dampen potential increase in false positive rate. To give examples, a name pair merged via misspelling condition is "Abdelaziz Bouteflika" and "Abdelaziz Bouteflicka". A name pair merged via near-duplicacy is "Muhsin al- Aini", "Muhsin al-Ayni (Aini)".

At the end, we have a unified name, name list (i.e. different names reduced into corresponding unified name), identifier list, and description list per person. In Section 3.1.1 Person Wikification, we will match each person with their Wikidata entries, if exist, and further do some record linkage on found entries.

Person Wikification

In our work, we use entity linking for matching persons with their corresponding Wikidata entries, if exist. Our entity linking method differs from the works mentioned in Section 2.2.2, because we already know the entity category (e.g. Q5 for persons), hence we do not need any textual context surrounding the entity, nor a scoring mechanism among candidate Wikidata entries.

Wikidata is queried using SPARQL. Given a person processed in Section 3.1.1 Person Unification, for each of that person's names in its name list, we search if a Wikidata entity exists (i.e. `?entity P31(is_instance) Q5(human)`). See Figure A.1 for corresponding SPARQL query. For some common names, multiple matches occur, along with the person we are looking for. In this case, we use Sentence Transformers (S-BERT) for comparing Wikidata person description against FRUS person description. First, we compute the average S-BERT embeddding of the person's FRUS descriptions. We use averaging because we observe that same person tends to have slightly different FRUS descriptions across volumes depending on the annotator's explanation. Later, we obtain S-BERT embeddings for each candidate Wikidata entry description. These descriptions tend to be short sentences about that person especially featuring role or occupation. Finally, we select the Wikidata entry whose embedding has highest cosine similarity with average FRUS embedding. Having linked persons with Wikidata, we reduce persons who are not unified during person unification but have the same Wikidata entries.

We can further fetch information from Wikidata for the persons who have Wikidata entry. With appropriate SPARQL queries, we extract gender, religion, school, occupation, positions held (i.e. role), citizenship, and political party membership. Refer to Figure A.2 for an exemplary SPARQL query for political party membership. Among them, position held, citizenship, and party membership include start and end dates, which we obtain as well. This is in line with the dynamic nature of our KG, where nodes and relationships have timestamps when available. In addition, we fetch country information for political party and school. Wikidata itself is not complete, therefore will not be ours. In Section 5, we report completeness statistics, and develop a solution for augmenting our KG.

Term Unification

We extract terms from every volume which has `<terms>` tag within `<front>`. Each term has a name, abbreviation, and volume-specific identifier. A term's scope is broad: it can be an institution (e.g. NATO), a common idiom (e.g. FY for fiscal year), a location (i.e. CV for Central Vietnam), and more. Similar to person unification, we reduce exactly matched names and concatenate their ids and abbreviations. Second, we reduce names with exactly same words but different combinations and concatenate their ids and abbreviations. The last step considers reducing obvious misspellings. We define misspelling between a name pair if its Damerau-Levensthein distance is at most 2. This is subjective, and it could be treated as hyperparameter.

At the end, we have a name, abbreviation list, id list per term. Here, a term mentioned in a document is a clue about its content. Hence, terms can be used as annotated keywords. We admit that term wikification is very challenging, as we do not know which entity to search for unlike in person wikification. Then, scope becomes whole Wikidata, and search based on just title matching returns many unrelated entries, requiring laborious post processing. Assumed this could be done, it is questionable if we should seek for a Wikidata entry for some terms such as alternating current (AA), oiler(vessel)(AO), or important question(IQ). For all the aforementioned reasons, we do not do term wikification.

City-Country Matching

In Section 3.1.1 Basic Extraction, we extracted source city for each document, available in free text. In this section, we aim to process free text, and match each city with its country, which would contribute another level of depth to our KG. In contrast to Section 3.1.1 Person Wikification, we do not know which entity in Wikidata to look for. Source city within `<placeName>` can be city, state, sub-country, specific area, or country itself, depending on the elaboration of document's reporting person. On the other hand, we cannot blindly query Wikidata for name match because of the overwhelming quantity of towns, and villages there. For example, there are many "Naples" in U.S., beyond the city in Italy in general search. Same happens with Chinese village, or Spanish-Latin American place names. For this, we designed a solution that is hugely automatized, but still requires manual check. This check is optional, and leads highly successful match, but can be time-consuming depending on aimed accuracy.

First, we split free text by comma. The second string (extension) usually is a general place name, while the first one is specific (i.e. Moscow, Russia). Then, we use a trusted city, sub-country, country database [45] [46] to match if the extension is either subcountry or country. If no extension exists, we search for the string itself as well. The purpose of this step is quickly matching strings that are supposed to be place names. Here, we observe that source cities mentioned in FRUS are usually important cities (e.g. capitals, big cities, war-time focal points), hence likely to appear in the mentioned database. At the end, several candidate countries might occur that needs to be resolved manually.

Second, for names with no city-country match in the first step, we query Wikidata with tight conditions: match occurs if place is capital city `Q5119`, or big city `Q1549591`. See Figure A.3 for the SPARQL query for searching if given city is capital. At the end of this step, manual check is done to detect any mismatches due to querying. We implemented code in a way that this and previous checks can be done at the same time.

Finally, we search for Wikidata entries of countries found in this process via `?country P31(is_instance) Q6256(country)`. With this, we can easily do country-wise analysis, and complement the KG with rich statistics. Note that we have not linked cities to their Wikidata entries as we have not deemed this necessary.

3.1.2 Enrichment

Named Entity Recognition

In FRUS corpus, only persons and institutions are annotated. However, nationality, country, organization or events mentioned in document may provide clues about content, and topic. We extract named entities with Spacy's NER [47]. We exclude date, time, quantity, ordinal, cardinal, money, and percent values. As some volumes do not have person and term annotations, NER can be used to extract these as well in future iterations.

Redaction Extraction

Omitted text that remains classified after declassification review is called redaction. In FRUS, redactions are in the format of italic string in between bracketed insertions within the document's text. For example `[<hi type="italic"> 1 line not declassified. <\hi>]`. The amount of redacted material has been noted by indicating number of lines or pages of source text. Redactions are important as they tend to hide most critical intelligence actions such as person, decision, money amount, or place name.

For each `<hi type="italic">` child of document tag whose text has "not declassified" sequence in between `<hi>` tags, we obtain that text. This is a semi-structured text, so not unified, but we want to obtain both redaction amount and type for better analysis. We say semi-structured because editors usually report redaction type within a small set of nouns such as line, paragraph, subparagraph, name etc. Plus, redaction text is usually duplicated with same information such as "5 lines (1/2 paragraphs) not declassified".

Considering these, we first compute type frequencies across all redactions using Spacy's POS tagger and count nouns. Type 'line' has the overwhelming majority. Then, if duplication occurs, we only proceed with most frequent type. For example, we reduce above example to "5 lines not declassified". Finally, we detect numerical expression which quantifies redaction type, with spacy's `like_num` function, and come up with a particular redaction's amount and type information separately, ready for analysis.

Topic Modeling

Topic modeling provide general view of semantic landscape, show most frequent topics, and interconnect documents across different years. We use both LDA and BERTopic in FRUS. For BERTopic, we use two flavors: one with original documents, another with named entities removed from the documents. Latter is considered because, in the former case, most important words to describe topics are usually geographical places, or important person names. By removing named entities we intend to obtain general topics agnostic to specific events. Topic count is 50 for LDA, whereas BERTopic has a higher limit (250), which enables to reveal more fine-grained details. In any case, we provide all 3 models in our KG. The flexibility is in that researchers can generate case-specific topics, and use our modular approach to add and analyse these via the KG.

Target Based Sentiment Analysis

In Section 2.1 Sentiment Analysis, we explain the difficulty of this problem and the lack of robust solutions. Here, we provide a solution [48] to enrich our KG even more. The polarity score (i.e. positive/negative) of a particular entity in a document is computed as the average polarity scores of adjectives occurring in the sentences that entity is present in. Adjectives are detected by Spacy, and word sentiment scores are obtained from *textblob* [49]. It is important to remark that this approach is error-prone, and should be interpreted cautiously. Still, in average, i.e. not at the level of individual instances, these results should capture interesting semantic trends. Lastly, we note that this method gives U.S. biased entity sentiment estimates, as the reporters are U.S. officials.

Expanded Schema

We have added various features to our KG since we have introduced it first in Section 3.1.1 Basic Extraction. These are person wikification, city-country matching, and enrichments. The expanded version of our KG is shown in part **B** of Figure 3.3. This schema does not show node attributes such as document length, lexical richness, person name, or topic details. For a complete list, please refer to `frus_conversion.py` in codebase.

3.2 Knowledge Graph Applications

We now have a huge graph-based dataset that hide many interesting patterns within its connections. In this section, we put forward several applications powered by node embeddings, centrality algorithms, and link prediction to uncover these patterns.

3.2.1 Dynamic Entity Embeddings

Word2vec's *"A word is known by the company it keeps."* philosophy marked a new era in NLP. Here, following the same idea, we propose dynamic (i.e. temporal) entity embeddings which allow historians to practice comparative history in scale.

Countries adjust their foreign policy in the course of history due to war, collaboration, economic interests, etc. Therefore, entities in FRUS are in constant movement, and this is reflected through documents. We think that an entity's shift can be obtained by reviewing most frequent entities it co-occur within different time slices. In other words, we say that frequently co-occurring entities are contextually similar. This alone can give clues about the content and reason of that entity's presence. Furthermore, combined with the enrichment options we

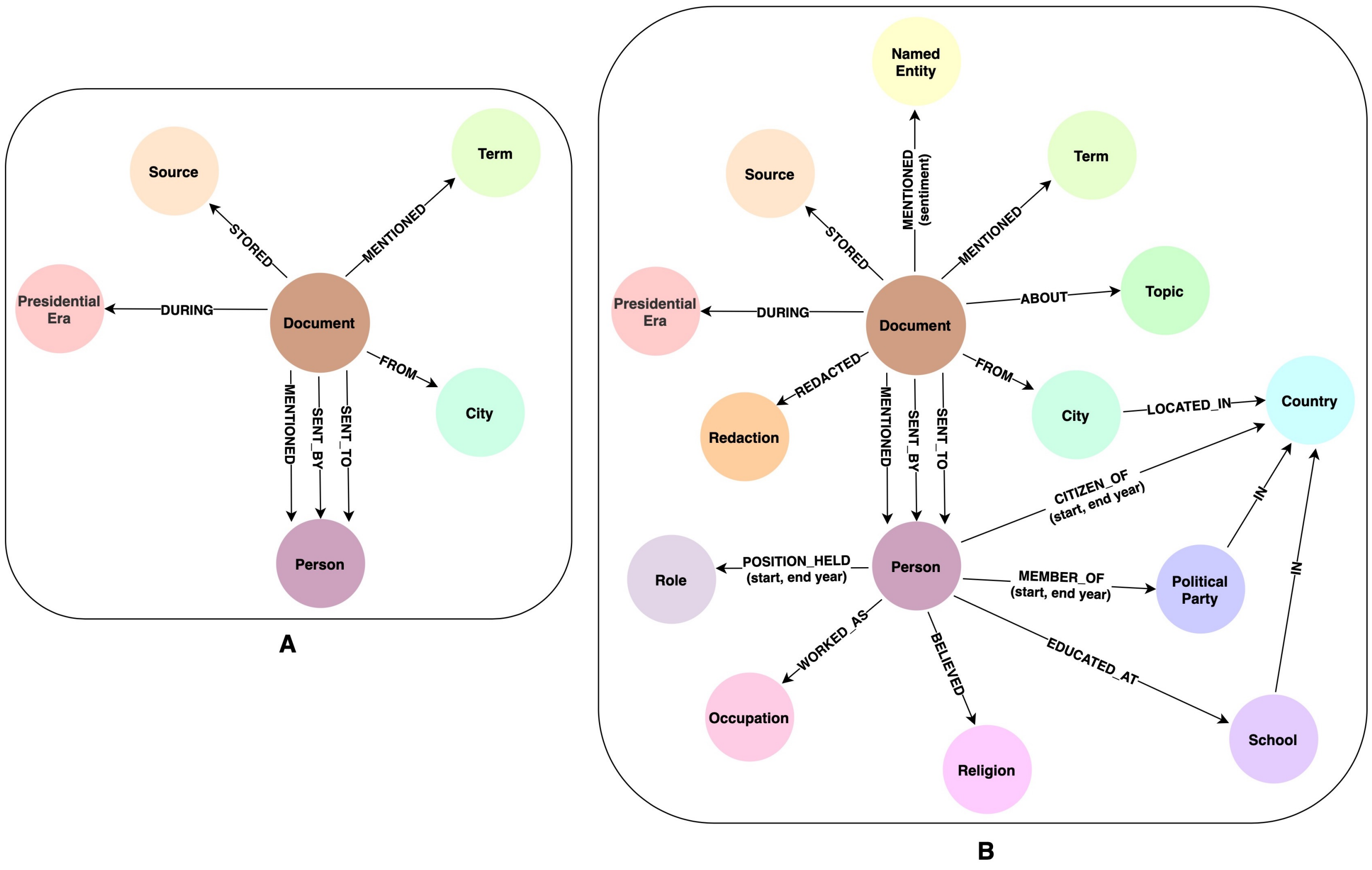


Figure 3.3: Base (**A**) and Expanded (**B**) Schema of our Knowledge Graph

provide, it enables full historical view instantly for any entity. Here, an entity can be person, institution, or one obtained from named entity recognition. Our pipeline is as follows:

1. We use named entities recognized in Section 3.1.2 Named Entity Recognition, and put a threshold on corpus-wide count to exclude infrequent entities.
2. We bin entities using the year of the documents they are mentioned in. Bin size can be adjusted. We use 4 years windows (e.g. Angola 62-66).
3. We created a relation (`NEIGHBOR`) in our KG between each co-occuring entity pair within a document. We set the relation weight to 1 for newly connected entities, and increase it by 1 for existing connections (that have already been connected in another document). We illustrate this process in Figure 3.4. This way, relation weight represents how frequently a pair

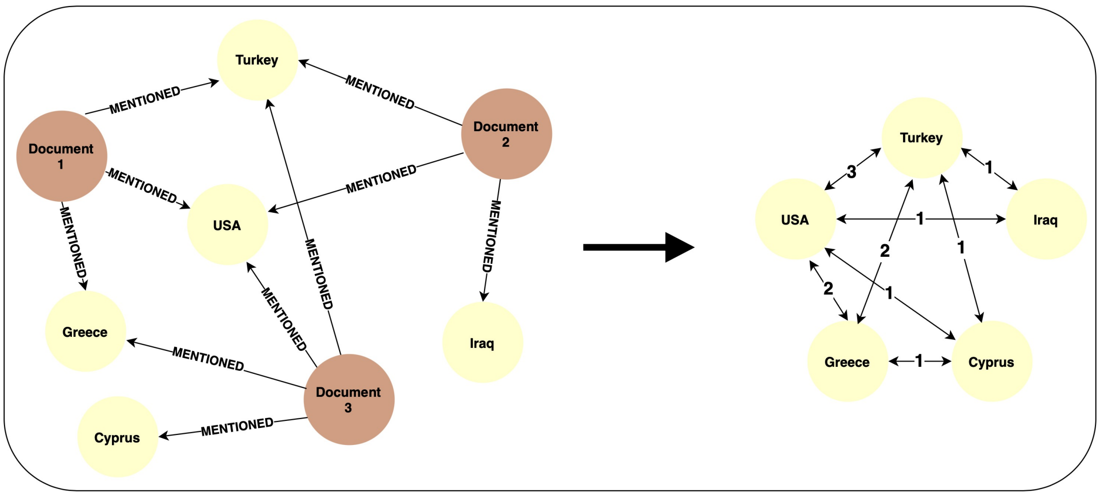


Figure 3.4: Conversion of document-entity mentions into homogeneous entity co-occurance graph

co-occurs for a certain time period.

4. Using this homogeneous entity graph, we apply node embedding methods (Section 2.2.4), namely FastRP and Node2vec, to obtain dynamic entity embeddings.

After these steps, we can start analysis. For example, we can obtain which entities are the most similar over time using cosine similarity between entity embeddings. We will further discuss this in Section 5.

3.2.2 Knowledge Graph Augmentation

Our KG is incomplete, as reported in Section 5. Incompleteness comes from two sources mainly: Wikidata is incomplete to provide religion, school, political party, etc. for all matched persons, plus our SPARQL query to find person names in Wikidata looks for an exact name match so misses some persons who have entry, but with a different denomination.

KG augmentation (i.e. link prediction) deals with this problem. Given that many `Person` nodes are not linked to Wikidata, we leverage this technique to try find as much of the missing information as possible. In FRUS, we have description for each annotated person, which usually describes role, occupation, and nationality. High similarity between FRUS descriptions might indicate same role, occupation, citizenship, party or religion information (nodes), and we use this as proxy information to perform link prediction. Our pipeline is as follows:

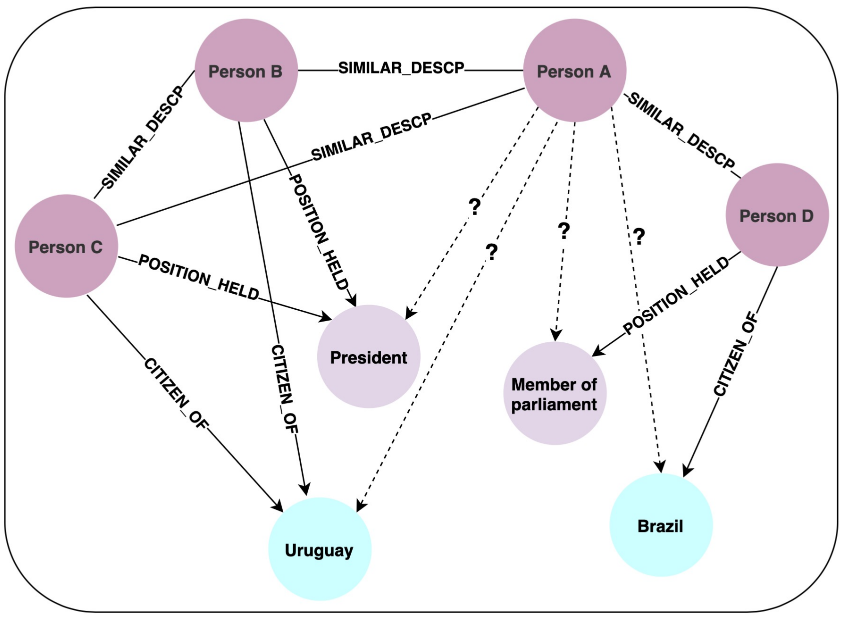


Figure 3.5: Similar FRUS descriptions give clues for identifying potential links.

1. We compute the average S-BERT embedding of each person's description, using the results from person unification.
2. We find 10 (hyperparameter) closest persons to each person in terms of cosine similarity, and create a relation (`SIMILAR_DESCP`) in our KG between these.
3. We project our KG depending on the task. For example, if we are going to predict missing `BELIEVED` relationships between `Person` and `Religion` nodes, we only include `Person`, `Religion` nodes, and `BELIEVED`, `SIMILAR_DESCP` relationships. Please see visualization provided in Figure 3.5.
4. We split `BELIEVED` relationships in the projected graph into feature set, training set, and test set.
5. We compute node embeddings by only using nodes connected by `BELIEVED` relationships in feature set, and all `SIMILAR_DESCP` relationships. Embeddings blend description similarity, and religion information smoothly.
6. We obtain relationship features of a node pair, which is computed as Hadamard product between that pair's previously computed node embeddings.
7. We train logistic regression model on `BELIEVED` relationships in training set where positive label means relation exists between a node pair, and vice versa. This way, the model understands what kind of feature pattern

drives into relation existence. For training, we need negative labels as well to counterbalance probability distribution, and Neo4j provides these from pairs which have not connected in projected graph.

8. Finally, we can use the trained model to predict new connections in our KG (i.e. religion information for persons without Wikidata entries). For a node pair, we compute Hadamard product between their node embeddings, then pass it through the model to get the probability of if a relation exists.

Steps 3-8 can be easily operated by Neo4j Link Prediction pipeline [50]. We report qualitative analysis on performance in Section 5. It is important to stress out that our solution should be treated more as a support to the job of human annotators, and mostly valid from the point of view of global results, but not individual cases. It works in scale so annotating from scratch takes more time than accepting or rejecting the model's prediction.

3.2.3 Role and Person Importance Scores

We devise a mechanism to find how central a `Person` or a `Role` in history is. We think that a person attains importance if he/she has been mentioned in diverse set of documents, so has lots of interconnections with other persons. For this, we connect persons mentioned within a document with each other in the same fashion described in Section 3.2.1. We use no relation weights, but it could be used as well. Finally, we run PageRank centrality algorithm described in Section 2.2.5 to obtain the importance score for each person.

People are temporary, roles are permanent. Hence, obtaining role importance and comparing in FRUS setting is noteworthy. We obtain role importance scores similar to person scores. The difference is that we only consider persons with a connection to a `Role` node, whose start and end dates exist. Each person's `Role` that was held in the date of a document is connected with other persons' `Role`s with the same criteria. Start and end years are valuable to compute this condition, and dictate that only the true role has significance. Now, we have a homogeneous graph of `Role` nodes, and run PageRank centrality algorithm again to obtain the importance score for each role. Dynamic role importance scores are also possible using the same binning technique mentioned in Section 3.2.1. We will argue which persons and roles are important in FRUS in Section 5.

Chapter 4

Statistics & Evaluation

In this section, we provide descriptive statistics about our KG, measure the performances of person unification and wikification, and finally give detailed information about our codebase and implementation.

4.1 Statistics of Knowledge Graph

To populate our KG, we use FRUS volumes from 1952 to 1988. We use a subset to show a proof-of-concept of our entire methodology. Whole FRUS should be used for a broader historical interpretation, but we advocate this span is sufficient to showcase the methods and applications intended with this thesis. Thus, all statistics below show the period ranging from 1952 to 1988.

Table 4.1 shows volume count, annotated volume count, and document counts. We note that some volumes are missing person and term annotations. Here it is not severe, but we observe that from 1940 backwards, nearly all volumes lack person annotations. Document count is nearly a quarter of whole FRUS document count, which ensures performing a broad historical analysis.

FRUS element	Count
Volume	238
Volume with person annotation	234
Volume with term annotation	238
Document	88016

Table 4.1: Volume, File, Annotation counts

Figure 4.1 displays volume distribution by presidential era, plus volume count with missing person annotations. We note that Truman and Reagan periods extend before 1952, and after 1988, respectively. Similar figure on term annotations by presidential era is in Table A.5.

Figure 4.2 shows document distribution by presidential era. We see similar pattern as in Figure 4.1.

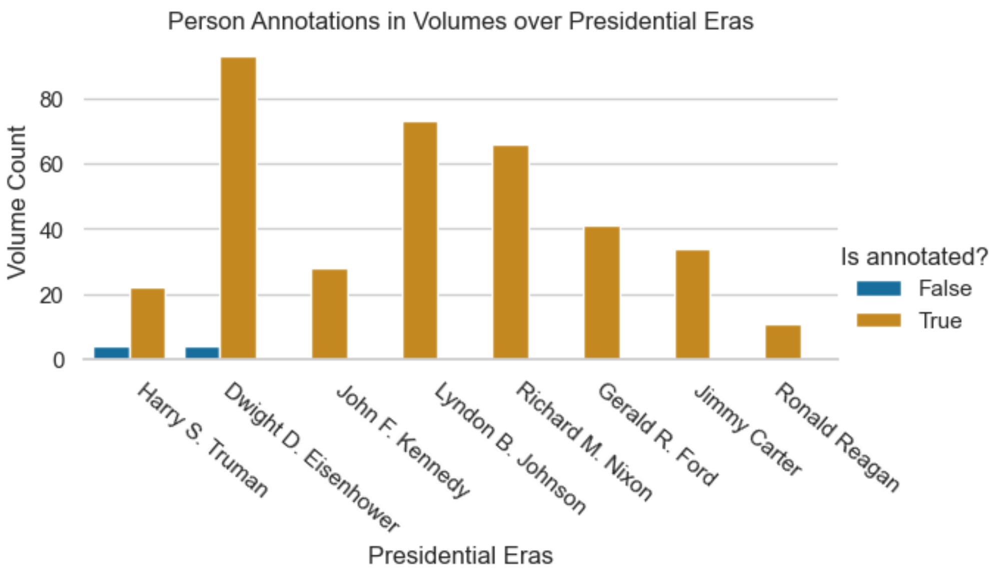


Figure 4.1: Volumes with person annotations over presidential eras

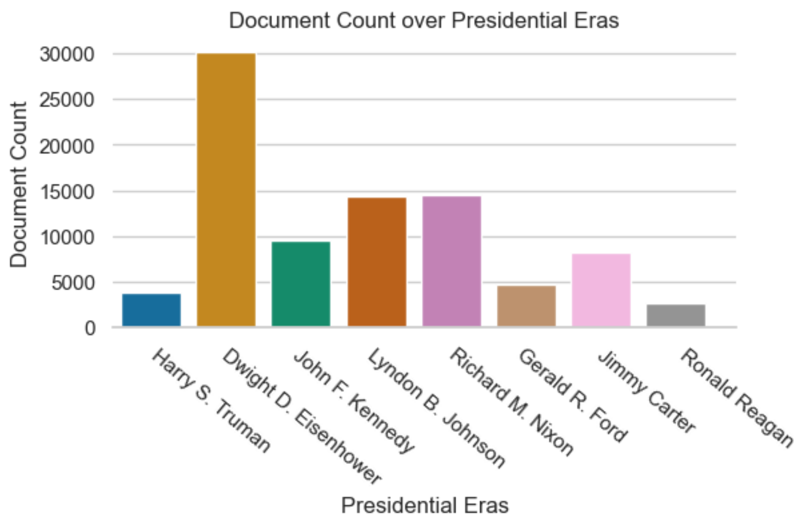


Figure 4.2: Document count over presidential eras

We described a person unification algorithm in Section 3.1.1 Person Unification. In Table 4.2, we show the unique person count after each step of this algorithm. It approximately merges 72% of the instances into others, unifying all entries into a more cohere set. Out of this, 60% comes from reducing exactly matched names. This actually shows that person annotations, after preprocessing, are tidy, and near-unified. A low hanging fruit is searching for names with exactly same words but different combinations, which reduces further 5%. After that, depending on how much we tolerate false positives, we can play with near-duplicate and typo detection hyperparameters to increase or decrease reduction effect. Here, with conservative hyperparameters, it reduces around 4000 names. A final step is checking for same Wikidata entries for different names. This step is meaningful because it finds names such as Nehru, and Indira Gandhi as the same person.

Person Unification Step	Person Count
Step 0: Extract person annotations from volumes	48363
Step 1: Reduce exactly matched names	19352
Step 2: Reduce names with exactly same words but different combinations	17633
Step 3: Reduce near-duplicate names & obvious misspellings	13317
Step 4: Reduce names with exactly same wikidata entries	13079

Table 4.2: Person unification statistics

We described a term unification algorithm in Section 3.1.1 Term Unification. In Table 4.3, we show the unique term count after each step of this algorithm. It approximately merges 74% of terms when done. A huge 72% of this reduction comes from reducing exactly matched names. As stated before, terms can be seen as already annotated keywords for understanding the document content.

Term Unification Step	Term Count
Step 0: Extract term annotations from volumes	50992
Step 1: Reduce exactly term names	14279
Step 2: Reduce names with exactly same words but different combinations	14157
Step 3: Reduce obvious misspellings	13034

Table 4.3: Term unification statistics

Person wikification (Section 3.1.1 Person Wikification) is done by searching all variations of a unified name within Wikidata. We might not find a person in Wikidata if that person is historically unimportant hence does not have an entry. This can be seen in Figure 4.3, where top most mentioned persons are linked to their Wikidata entries with high success. Success gradually diminishes when we include infrequent persons. This is relieving as we expect most mentioned persons to be most historically important ones as well. Parallel to this outcome, we note that out of all the mentions to a person in all documents, 82% have Wikidata coverage (315805 in 386342), which is underlining the validity of this method for further historical analysis.

Our KG has many node and relationship types as shown in the schema in Figure 3.3. In Table 4.4, we present count statistics for each node and relation type. Total number of node and relation counts show that even a subset of FRUS

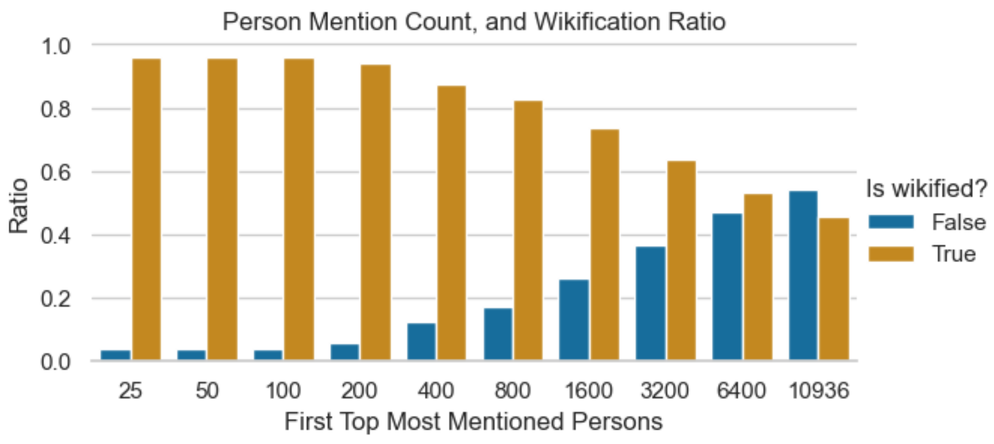


Figure 4.3: person mention count vs wikidata match

corpus leads to a huge KG.

Knowledge Graph Node Type	Node Count
`Document`	88016
`Presidential Era`	26
`City`	532
`Country`	311
`Term`	13034
`Person`	13079
`Role`	2652
`Occupation`	659
`Religion`	90
`School`	2237
`Political Party`	612
`Redaction`	27468
`NamedEntity`	4304
`4y_DynamicNamedEntity`	30335
`TopicBertWithEntities`	250
`TopicBertNoEntities`	100
`TopicLDANoEntities`	50
Total	149116

Knowledge Graph Relationship Type	Relationship Count
`DURING`	88016
`FROM`	78925
`LOCATED_IN`	422
`SENT_BY`	48090
`SENT_TO`	15320
`MENTIONED (Person)`	386342
`MENTIONED (Term)`	276493
`ABOUT (3 Topics)`	88016*3
`BELIEVED`	947
`WORKED_AS`	10357
`POSITION_HELD`	8590
`MEMBER_OF`	2872
`CITIZEN_OF`	5324
`EDUCATED_AT`	7141
`IN (School)`	2206
`IN (Pol Party)`	608
`REDACTED`	27468
Total	1223041

Table 4.4: Entity and Relationship Counts in our Knowledge Graph

4.2 Methods Evaluation

4.2.1 Person Unification Performance

We measured our pipeline's unification performance by randomly selecting 423 unified names, and manually labeling these as mismatch or correct. For a person, given a list of names, it is correct if all names refer to same real person, and mismatch otherwise. For mismatches, we also count number of different persons wrongly unified into one person. We also include the mention count per unified name for investigation.

Correct unification is critical as high error presence might alter the analysis. In Table 4.5, we present different statistics to assess this. Binary accuracy (i.e. if all unified names is same person) is 91.2%. From table, we observe that mismatches tend to occur for longer lists unified into single name. This is expected as the names in FRUS tend to include roles such as Major, or General, hence could not get filtered fully. Please note that mention count is high for mismatches because wrongly merging names into a single entry increases the mention count. Overall, we are satisfied by the performance. Given that these validation results are manually obtained, it is not possible to play with other configurations to find the optimal result.

		List Length	List Length	Person Count	Person Count	Mention Count	Mention Count
Label	Count	Mean	St Dev	Mean	St Dev	Mean	St Dev
Mismatch	37	4.70	3.20	3.03	1.71	131.43	243.45
Correct	386	2.79	1.26	1.00	0.00	70.64	203.79

Table 4.5: Person Unification Performance. List length is the count of names unified into single name. Person Count is actual real person count in a supposed-to-be unified name. 1 for correct, more than 1 with max of List Length for mismatch. Mention Count is person mention count for that unified name.

4.2.2 Person Wikification Performance

We measured our pipeline's wikification performance by randomly selecting 200 names (100 with single Wikidata candidates, 100 with multiple Wikidata candidates), and manually labeling these as mismatch or correct. Correct wikification is critical as we expand the graph with Wikidata entities such as `Role`, `Religion` or `Citizenship`, and utilize them in further downstream tasks. Hence, we should be cautious about introducing errors now, that could further propagate. In Table 4.6, we present the performance on persons with single candidate Wikidata entries. Accuracy is 90%. Statistics about person mention counts tell that our algorithm tends to mismatch entries for less important persons (i.e., which appear less frequently in the text), which is a desirable result for the sake of consequent explorations.

Label	Count	Mean	St Dev	min	25%	50%	75%	max
0	10	28.00	80.88	0	0.25	1.0	6.50	258
1	90	48.38	197.15	0	2.00	5.0	17.25	1796

Table 4.6: Person Wikification Performance Single Candidate Entry. All statistics except Count is Person Mention Count.

In Table 4.7, we present the performance on persons with multiple candidate Wikidata entries. As we can observe, the accuracy is 81%, and the statistics about person mention counts tell that our algorithm tends to mismatch entries with less important persons, which is positive.

Label	Count	Mean	St Dev	min	25%	50%	75%	max
0	19	19.11	36.14	0	1.0	2.0	25.5	147
1	81	54.06	165.88	0	2.0	6.0	28.0	1324

Table 4.7: Person Wikification Performance Multiple Candidate Entry. All statistics except Count is Person Mention Count.

In general, almost all mismatches stem from the fact that our algorithm is deterministic. This means that it still gives a match if the person we are looking for does not have an entry, but an entry with exact same name exists (frequent for common names). Further improvements such as using S-BERT for a probabilistic matcher, enabling to stay silent for these cases, can be implemented in the future.

4.3 Implementation Details

We use Python 3.8.8 for parsing, enrichment and the population of the KG. We use Neo4j (5.2.0), a native graph database to store and query our KG. Native graph databases are designed to both store and process data as a graph. They can navigate fast through long chain of connections without the overhead of index lookup, or many join operations. For example, Cypher, the querying language of Neo4J, enables swift and intuitive querying, leveraging the relations. On contrast, non-native graph databases store nodes and relations as unrelated entities, which might be stored far apart in memory. That requires another layer of work to interpret data as graph, and decreases speed and efficiency. To populate with all information obtained through parsing and enrichment, stored as pandas dataframes, a Neo4J KG, we use LOAD CSV function of Cypher [51]. Below, we elaborate which file in codebase correspond to which section in this report.

1. `eda.ipynb` file captures Section 3.1.1 XML Structure Discovery.
2. `constants.py` stores necessary constants used throughout parsing, enrichment, and KG population.
3. `document_extraction.py` captures Section 3.1.1 Basic Extraction.
4. `person_unify.py` covers Section 3.1.1 Person Unification, and Section 3.1.1 Person Wikification (only searching for entity).
5. `extract_person_extras.py` handles Section 3.1.1 Person Wikification (fetching extra information such as religion, gender, etc of person, if found).
6. `term_unify.py` covers Section 3.1.1 Term Unification.
7. `city_country_extraction.py` captures Section 3.1.1 City-Country Matching.
8. `redaction_extraction.py` is for Section 3.1.2 Redaction Extraction.
9. `lda_topic_extraction.py` and `bert_topic_extraction.py` files are for Section 3.1.2 Topic Modeling.
10. `extract_entity_sentiments.py` file captures Section 3.1.2 Target Based Sentiment Analysis.
11. `extract_entity_bins.py` extracts and bins named entities in texts, that is first two steps of Section 3.2.1.
12. `link_prediction.py` finds most similar persons on FRUS descriptions for each person, that is first two steps of Section 3.2.2.
13. `frus_conversion.py` stores a conversion plan from csv to Neo4j suitable to the schema of our KG.
14. `cypher_commands.txt` is the compilation of different cypher commands to obtain results for Section 5.2, Section 5.3, Section 5.4 and Section 5.5.

Chapter 5

Discussion

In this section, we aim to spark inspiration on what is possible with FRUS KG. We approach only from the computational perspective, and skip giving further comments from the point of view of the history, as that is not the aim of the thesis. We believe that further systematical and richer analysis can provide extended political science insights.

5.1 Basic Analysis

The world map in Figure 5.1 shows document distribution by countries. We count a document from a particular country if the source city the document was sent from is located in that country. We exclude documents sent from U.S. because it has huge majority, hampering visualization. We note high activity in Europe, and moderate activity in Japan, Russia, Middle East, and Caribbean.

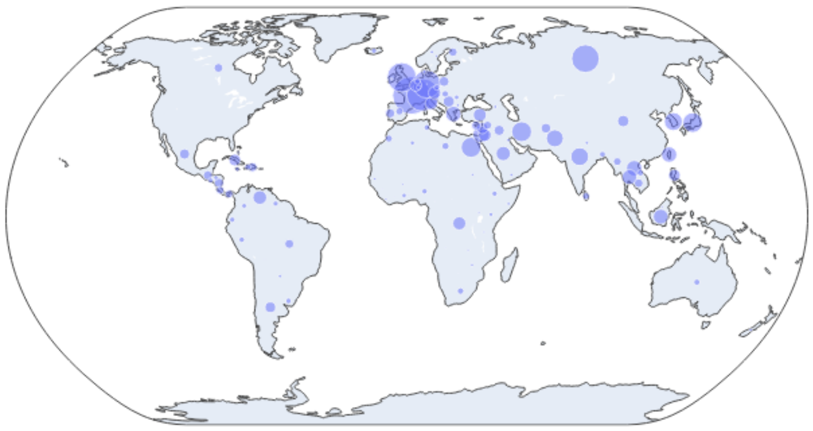


Figure 5.1: Document Distribution by Country (excluding U.S.)

By aggregating countries into continents and binning according to presidents, a clearer picture emerges. In Figure 5.2, we see which continents raised a lot of attention during different periods. Please note that we exclude documents originating from U.S. itself. Here, we see that President Johnson has many documents from Asia, which is probably due to Vietnam War, and President Carter has unusually many documents from Africa, which is probably due to his stance against white minority rule in South Africa and Rhodesia [52].

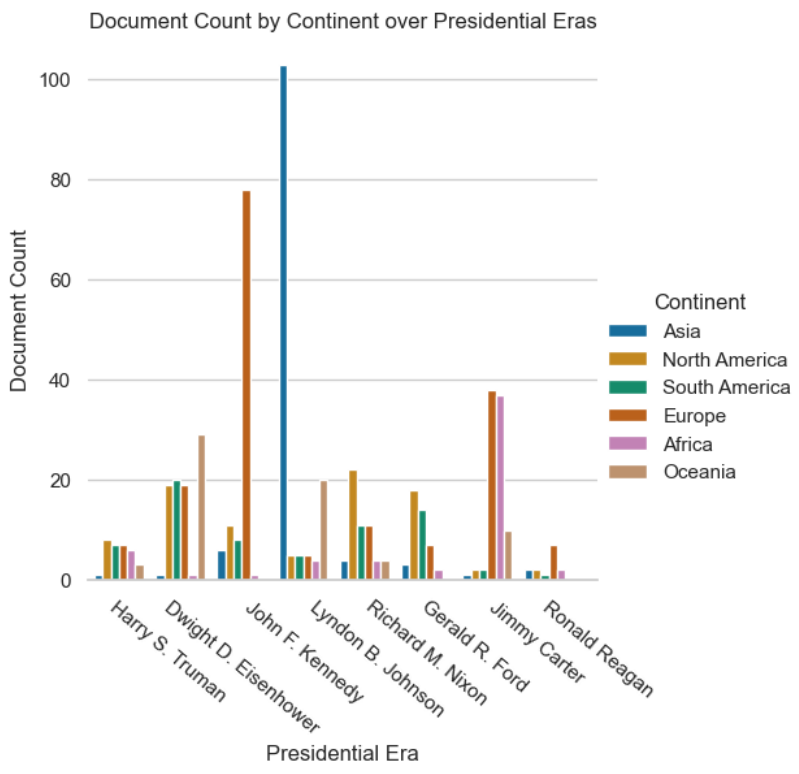


Figure 5.2: Document Count by Source Continent over Presidential Eras (excluding documents originated from U.S)

Another interesting question is if the texts have became longer and more complex over time. We answer this in Figure 5.3, where we measure complexity by corrected type token ratio (CTTR), and length by token count. We see that document length tends to increase over time. CTTR increases with length only if the word diversity increases even more. Here, the two follows similar patterns, thus we can conclude that word diversity tends to increase over time as well.

Each period comes with own political affairs. Even though the general topics (i.e. war, economy) remain the same across time, location and details change. For this reason, we use two different BERT topic modeling: one with named entities, and one without them. Former captures details, while the latter captures general concepts. We can see some examples of this in Table 5.1. All presidents are busy with military, and political positions, while a close inspection shows that locations change from Africa, to Russia or Cuba.

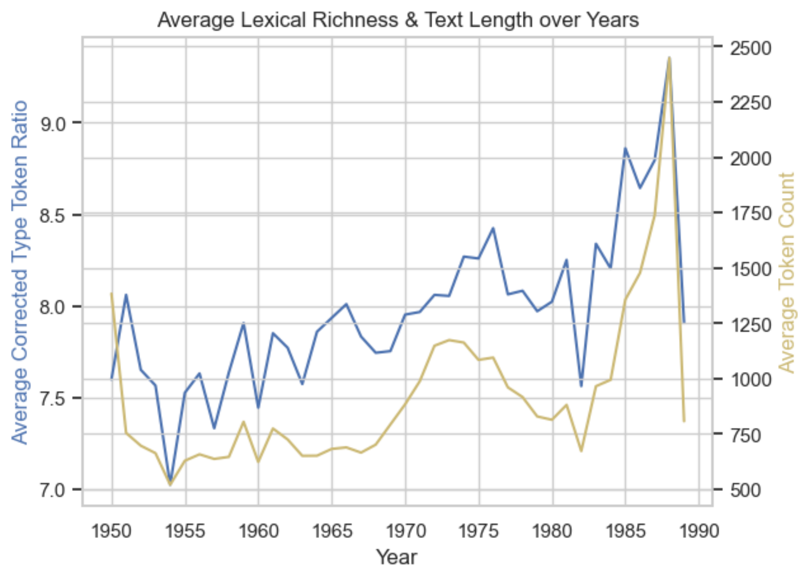


Figure 5.3: Average Lexical Richness and Token Count over Years. Lexical richness measured by corrected type token ratio (CTTR).

President	Most Frequent Topic	Document Count
Harry S. Truman	38_mosadeq_oil_compensation_consortium 3_intelligence_elections_molotov_support	178 352
Dwight D. Eisenhower	0_arab_algeria_saudi arabia_jordan 0_military_political_policy_position	1258 2938
John F. Kennedy	3_tshombe_ethiopia_congolese_lumumba 0_military_political_policy_position	533 1667
Lyndon B. Johnson	3_tshombe_ethiopia_congolese_lumumba 0_military_political_policy_position	622 2938
Richard M. Nixon	21_unclear_going_know_haig 0_military_political_policy_position	495 1906
Gerald R. Ford	1_somoza_castro_peru_marcos 0_military_political_policy_position	221 572
Jimmy Carter	1_somoza_castro_peru_marcos 0_military_political_policy_position	571 1255
Ronald Reagan	42_falklands_galtieri_malvinas_secretary haig 0_military_political_policy_position	345 558

Table 5.1: Most frequent topics by presidents. Top row: BERTopic with named entities. Bottom row: BERTopic without named entities.

5.2 Redaction Analysis

Even though a redacted passage is still classified, the content of its surrounding text can show which topics are most sensitive for intelligence purposes. Here, thanks to our enrichment efforts, we can trace both the redaction amount and type, and entities and topics involved in it. Figure 5.4 shows total redaction count per years. We observe a sharp drop in count around 1980, but we prefer to leave its interpretation to historians.

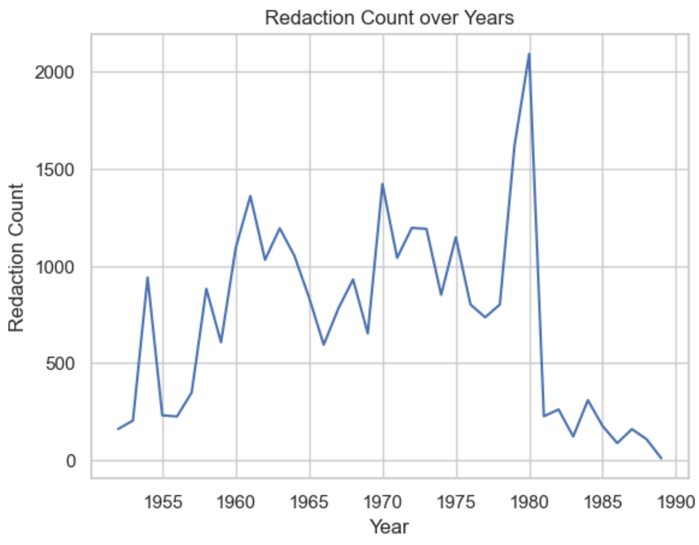


Figure 5.4: Shift in Redaction Count over Years

Figure 5.5 show frequency of redaction types on log scale. Here, `line` type has the overwhelming majority, and it is usually accompanied by a numerical value (e.g. 2 lines). Type `text` is a general way of indicating a redaction, `name` would be an interesting subject of study, as it is likely hiding persons with the highest attention to U.S. services. Finally, `dollaramount` type signs which topics have secret economical aspects.

Now, we aim to provide some exemplary analysis on redactions. Table 5.2 shows the most redacted topics according to BERTopic without entities. Naturally, they focus on military, intelligence, and war. Topic in second row seems to capture the main terms related to redactions, which can be seen as a validation of our methodology.

Table 5.3 shows the most redacted topics in terms of economical aspect. Intelligence officials might not want to reveal amount of money invested for certain policies, hence leave these fields classified. Entities mentioned in topic names, and possible events related to them, are likely to explain the reasons for this behaviour, as most of these topics are related with events occurring in third

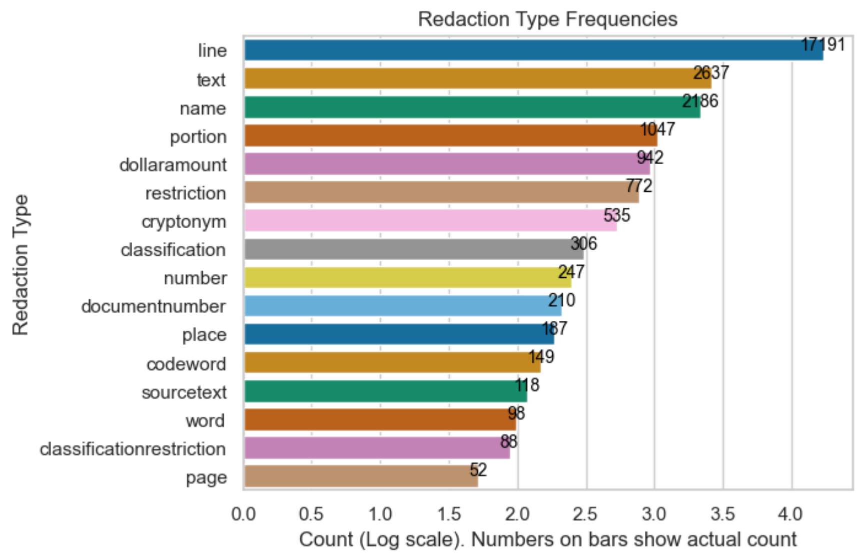


Figure 5.5: Redaction Type Frequencies on Log10 Scale. (Numbers on bars indicate actual type count.)

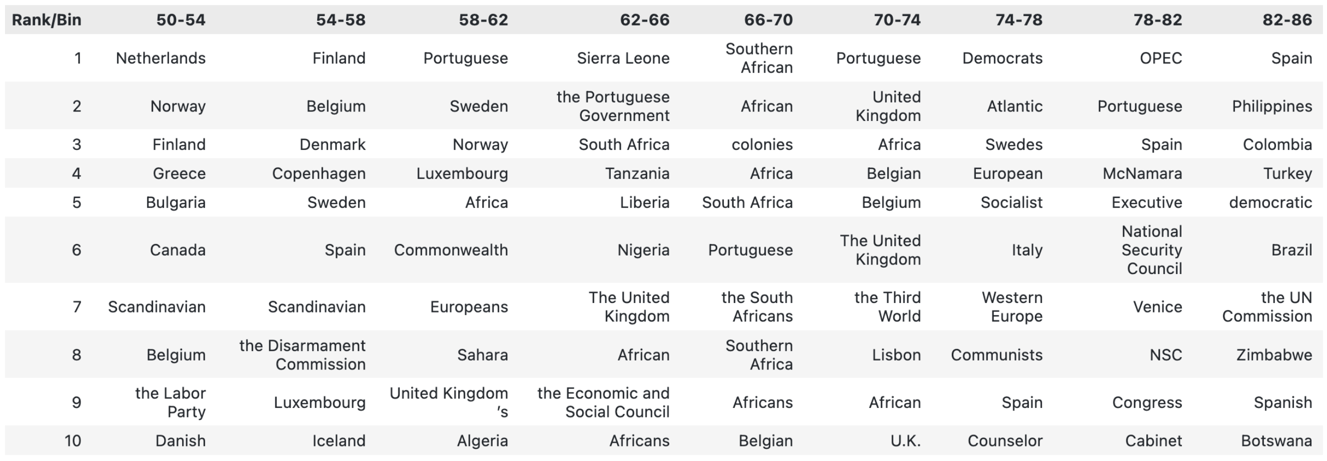
BERTopic without Entities	Document Count	Redaction Amount
"0_military_political_policy_position"	847	8076.5
"12_line_text declassified_source text_source"	377	4750.0
"3_intelligence_elections_molotov_support"	163	2130.5
"5_test ban_stockpile_agreement_safeguards"	94	1768.5
"1_ambassador_mr_cyprus_say"	216	1707.0

Table 5.2: Most Redacted Top 5 Bert Topics, measured by sum of redacted lines

countries, affecting their national policies. Please see Figure A.4 for the implemented Cypher query for this analysis.

BERTopic with Entities	Document Count	Redaction Count
"27_chilean_frei_pdc_dollar declassified"	55	454
"3_tshombe_ethiopia_congolese_lumumba"	39	105
"28_brezhnev_secretary kissinger_let_adm moorer"	10	86
"182_afm_pcp_azevedo_covert action"	14	34
22_dci_usia_community_activities"	8	22

Table 5.3: Most Redacted Top 5 Bert Topics in terms of dollar amount redaction



Rank/Bin	50-54	54-58	58-62	62-66	66-70	70-74	74-78	78-82	82-86
1	Netherlands	Finland	Portuguese	Sierra Leone	Southern African	Portuguese	Democrats	OPEC	Spain
2	Norway	Belgium	Sweden	the Portuguese Government	African	United Kingdom	Atlantic	Portuguese	Philippines
3	Finland	Denmark	Norway	South Africa	colonies	Africa	Swedes	Spain	Colombia
4	Greece	Copenhagen	Luxembourg	Tanzania	Africa	Belgian	European	McNamara	Turkey
5	Bulgaria	Sweden	Africa	Liberia	South Africa	Belgium	Socialist	Executive	democratic
6	Canada	Spain	Commonwealth	Nigeria	Portuguese	The United Kingdom	Italy	National Security Council	Brazil
7	Scandinavian	Scandinavian	Europeans	The United Kingdom	the South Africans	the Third World	Western Europe	Venice	the UN Commission
8	Belgium	the Disarmament Commission	Sahara	African	Southern Africa	Lisbon	Communists	NSC	Zimbabwe
9	the Labor Party	Luxembourg	United Kingdom 's	the Economic and Social Council	Africans	African	Spain	Congress	Spanish
10	Danish	Iceland	Algeria	Africans	Belgian	U.K.	Counselor	Cabinet	Botswana

Figure 5.6: Most Similar 10 Entities to Portugal over Years. Rank shows most similar to least. Bin shows the year span.

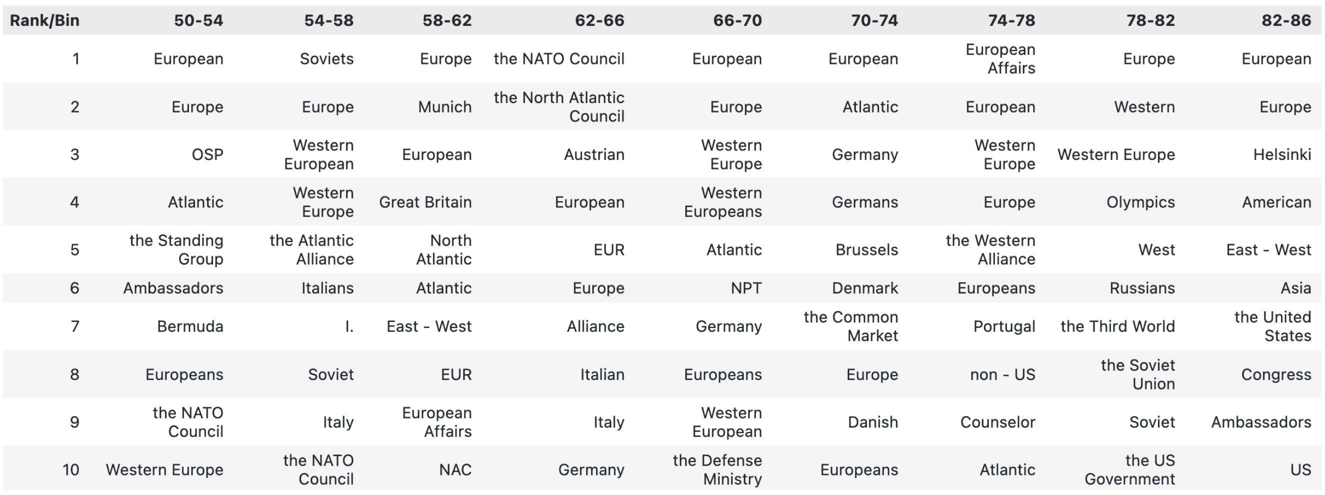
5.3 Analysis on Dynamic Entity Embeddings

We proposed dynamic entity embeddings, aiming at capturing how political entities evolved through time, getting closer or further away, as different historical and socioeconomic events unfold. If holds, we can naturally monitor important events from a computational point of view. In Figure 5.6, we present most similar entities over time to Portugal. Busy with Europe initially, it shifts its attention to African colonies during early 1950s and 60s, probably due to decolonisation era. Afterwards, it mostly deals with global issues such as communism, and OPEC. These comments, although helpful for validating the methodology, cannot go beyond, and would require a more profound analysis from a domain expert to fully unveil their value.

Dynamic entity embedding method aims to find contextually similar entities. It does not assume any prior knowledge, hence is an unbiased vision of the content of the FRUS. For example, the results in Figure 5.7, and how they align with historical events, serve as validation of such methodology. This figure shows the most similar entities to NATO over time. Although European terms dominate the top ranks, change in content appears in low ranks such as the presence of Standing Group in 50-54, and Common Market in 70-74. We present a similar analysis for Gibraltar in Figure A.6.

5.4 Analysis on Knowledge Graph Augmentation

Our KG is incomplete. In Table 5.4, we report statistics on this issue. Especially, `Religion` and `School` fields are severely sparse. To tackle this problem, we



Rank/Bin	50-54	54-58	58-62	62-66	66-70	70-74	74-78	78-82	82-86
1	European	Soviets	Europe	the NATO Council	European	European	European Affairs	Europe	European
2	Europe	Europe	Munich	the North Atlantic Council	Europe	Atlantic	European	Western	Europe
3	OSP	Western European	European	Austrian	Western Europe	Germany	Western Europe	Western Europe	Helsinki
4	Atlantic	Western Europe	Great Britain	European	Western Europeans	Germans	Europe	Olympics	American
5	the Standing Group	the Atlantic Alliance	North Atlantic	EUR	Atlantic	Brussels	the Western Alliance	West	East - West
6	Ambassadors	Italians	Atlantic	Europe	NPT	Denmark	Europeans	Russians	Asia
7	Bermuda	I.	East - West	Alliance	Germany	the Common Market	Portugal	the Third World	the United States
8	Europeans	Soviet	EUR	Italian	Europeans	Europe	non - US	the Soviet Union	Congress
9	the NATO Council	Italy	European Affairs	Italy	Western European	Danish	Counselor	Soviet	Ambassadors
10	Western Europe	the NATO Council	NAC	Germany	the Defense Ministry	Europeans	Atlantic	the US Government	US

Figure 5.7: Most Similar 10 Entities to NATO over Years. Rank shows most similar to least. Bin shows the year span.

propose to augment our KG, especially for the persons with missing Wikidata entries. We stated before that we use FRUS person descriptions as a proxy, and these descriptions are mostly about role, occupation, and citizenship of the person described. In this section, we aim to qualitatively assess if our design is promising. We report our model's predictions about missing Person-Citizenship information in Table 5.5, and missing Person-Role information in Table 5.6. Note that these examples are the ones our model puts highest probabilities, hence are most likely correct matches.

Wikidata Field	Unique Person Count	Ratio to Total Person Count	Total Matches
`Gender`	5444	41.6%	5444
`Religion`	891	6.8%	946
`Citizenship`	4532	34.7%	5305
`Occupation`	5017	38.4%	10289
`Role`	3120	23.9%	8545
`Political Party`	2331	17.8%	2862
`School`	354	2.7%	7096

Table 5.4: Knowledge Graph Completeness Statistics. Wikidata Field is the information piece we extract from Wikidata, and use in knowledge graph either as entity, relation or attribute. Unique Person Count is how many unique person has the corresponding field. Total Matches is total number of information piece in knowledge graph as a person might hold multiple information from the same field (e.g. two Occupations)

Looking at the tables, we observe that our model makes predictions in line with our assumption, which is that persons with similar FRUS descriptions should have similar Wikidata information. For example, it consistently predicts military personnel (which is apparent from person names) as Commandant of the Marine Corps. These persons might not hold that exact role, but in principle, our model captures correctly that these persons are likely to have roles from military. Hereby, we note that we care mostly about generalization ability of our model, rather than guessing each missing information one by one without error. This ensures our principle of creating scalable and automatic pipelines, that can be a support for human annotators. Since we deem the results promising, in further iterations we plan to include quantitative results as well.

Person	Prediction
Gudmundar Gudmundsson J.	Iceland
Figueres-Ferrer José Pepe	Costa Rica
Gertruda Sekaninova	Czechoslovakia
Hermann Jonasson	Iceland
Halldór Ásgrímsson	Iceland
Mulcahy R.	Irish Free State
Gonzalo González Solórzano	Costa Rica
Bashev Ivan	Bulgaria
Hernán Siles Zuazo	Bolivia

Table 5.5: Top 9 Highest Probability Predicted Links between Person-Citizenship. Through manual validation, all predictions been found correct.

Person	Prediction
Figueres-Ferrer José Pepe	President of Costa Rica
Mod Peter	member of the National Assembly of Hungary
Hont Janos	member of the National Assembly of Hungary
Gen. M. Pate Randolph	Commandant of the Marine Corps
C. General Lemuel Shepherd	Commandant of the Marine Corps
General J. Lieutenant McCaul Verne	Commandant of the Marine Corps
Kelley P.X.	Commandant of the Marine Corps
General Maleter Pal	member of the National Assembly of Hungary
General Greene Jr. M. Wallace	Commandant of the Marine Corps

Table 5.6: Top 9 Highest Probability Predicted Links between Person-Role

5.5 Analysis on Role and Person Importance Scores

The algorithm that we propose for computing node importance scores inherently utilizes overall connectivity of historical actors, as explained in Section 3.2.3.

This notion points out one of the major targets we hope for in this thesis, which is inspecting FRUS corpus as a whole, beyond individual documents. While we wait for interesting patterns obtained through the algorithm, we are already biased towards which roles can be more important. Actually, this might help us to validate our pipeline. In Table 5.7, we present most prominent 10 roles for 1952-1988 span according to PageRank centrality algorithm. As expected, we see that the top U.S. officials are on top, because they are senders, receivers, and decision-makers in the U.S. foreign policy. We believe that the U.S. president comes after the U.S. Secretary of State because the latter is the focal point of document traffic.

Rank	Role	Country	PageRank Importance
1	United States Secretary of State	U.S.	16.14
2	President of the United States	U.S.	12.99
3	National Security Advisor	U.S.	11.50
4	United States Secretary of Defense	U.S.	7.61
5	Under Secretary of State for Political Affairs	U.S.	7.49
6	General Secretary of the Communist Party of the Soviet Union	Russia	7.21
7	Assistant Secretary of State for European and Eurasian Affairs	U.S.	6.59
8	Chairman of the Joint Chiefs of Staff	U.S.	6.42
9	Federal Chancellor of Germany	Germany	6.14
10	Prime Minister of the United Kingdom	U.K.	6.13

Table 5.7: Top 10 Most Important Roles, and Corresponding Countries in FRUS

Similar to role importance, we can analyse person importance scores. In Table A.1, we present most prominent 10 persons in FRUS. We expect a correlation between top roles, and their holders, and results validate this, as the persons who has been Secretary of State, or President dominate the table. A remarkable point is that A. Henry Kissinger is at the top because he has held two very important roles in his career, hence leading to a huge connectivity in FRUS. This observation could only be possible via this type of computational approaches, harnessing the full data as a whole through the KG.

Chapter 6

Conclusion

In this work, we propose a new methodology for better handling and analyzing text corpora that are rich in entity-relation notions, such as the ones in political sciences. Main idea of this methodology is to utilize KGs to store free-form knowledge presented in the text. We craft a schema appropriate to the corpus and necessary tasks in hand, use various parsing and NLP techniques to extract these knowledge from documents, and finally convert extracted fields to the actual nodes and edges of the KG. By using U.S. foreign policy documents, we demonstrate the validity and extent of each of these steps, and obtain a political KG at the end. We further use this KG for historical analysis, by employing varied methods from simple count based statistics to advanced machine learning algorithms such as node embeddings, link prediction, and centrality methods. We believe that this computational approach can provide a new perspective in political sciences, and fields alike.

Limitations. Let us mention some hardships we face during our work, as well as our work's downsides. First, we do not utilize whole FRUS corpus from 1861 to 1998. Second, as we have to manually evaluate person unification and wikification algorithms, we may not find the best set of hyperparameters for these. In city-country matching, a manual verification of the found matches is required, although optional, for a more realistic KG, but this is the only part that prevents a fully automatic pipeline. In this work, we could not quantitatively evaluate the city-country matching algorithm, NER and link prediction model, although we provided qualitative evaluation emphasizing their functionality. Furthermore, for obtaining role and importance scores, we could extend it to be dynamic to allow evolving importance scores over time for each role and person. In discussion, we provide results to the extent that they are sufficient for explaining the power of their corresponding methods. This way, it may be perceived primitive from a social science perspective. Without doubt, historians would have better questions that are worthy of analysis, and should use our methods and suggestions to answer these.

Future Work. In future, we plan to work on improving our paradigm to be applicable to any corpora seamlessly. For example, our parsing, schema and extracted fields are specific to FRUS, but a general approach focused on entity-relation notions that could be automatically extracted by NER and alike would increase generality. From FRUS point of view, we will use the whole corpus to obtain a complete KG and better insights. Then, we will add quantitative evaluations for city-country, NER, and link prediction models, as we did for person unification and wikification in the current iteration. In addition, for city-country matching, we might propose an improved and fully automatic algorithm, such as applying its second step to extensions (second strings) as well instead of just first strings in the `<placeName>` tag. Furthermore, we will use NER to find persons and terms that are not annotated in earlier volumes, as well as richer analysis using provided `Terms` and `Named Entity` sentiments. At the end, we imagine an ultimate ready-to-use FRUS KG, which can be directly utilized by historians as a tool.

Bibliography

[1] T. Office of the Historian, "Foreign Relations of the United States," https://history.state.gov/historicaldocuments.

[2] "Introduction to Cypher." [Online]. Available: https://neo4j.com/docs/getting-started/cypher-intro/

[3] D. Vrandečić and M. Krötzsch, "Wikidata: A Free Collaborative Knowledgebase," *Commun. ACM*, vol. 57, no. 10, p. 78–85, sep 2014. [Online]. Available: https://doi.org/10.1145/2629489

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A Nucleus for a Web of Open Data," in *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ser. ISWC'07/ASWC'07. Berlin, Heidelberg: Springer-Verlag, 2007, p. 722–735.

[5] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 1247–1250. [Online]. Available: https://doi.org/10.1145/1376616.1376746

[6] "What is a knowledge graph?" [Online]. Available: https://www.ibm.com/topics/knowledge-graph

[7] B. Abu-Salih, "Domain-specific knowledge graphs: A survey," *Journal of Network and Computer Applications*, vol. 185, p. 103076, 2021.

[8] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," *Advances in neural information processing systems*, vol. 32, 2019.

[9] L. Shen, "LexicalRichness: A small module to compute textual lexical richness," 2022. [Online]. Available: https://github.com/LSYS/lexicalrichness

[10] J. R. Finkel, T. Grenager, and C. D. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL'05)*, 2005, pp. 363–370.

[11] X. Wang, Y. Jiang, N. Bach, T. Wang, Z. Huang, F. Huang, and K. Tu, "Automated concatenation of embeddings for structured prediction," *arXiv preprint arXiv:2010.05006*, 2020.

[12] E. F. Sang and F. De Meulder, "Introduction to the Conll-2003 shared task: Language-independent named entity recognition," *arXiv preprint cs/0306050*, 2003.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[14] D. M. Blei and J. D. Lafferty, "Dynamic Topic Models," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 113–120.

[15] M. Grootendorst, "Bertopic: Neural topic modeling with a class-based TF-IDF procedure," *arXiv preprint arXiv:2203.05794*, 2022.

[16] M. Saeidi, G. Bouchard, M. Liakata, and S. Riedel, "SentiHood: Targeted Aspect Based Sentiment Analysis Dataset for Urban Neighbourhoods," *CoRR*, vol. abs/1610.03771, 2016. [Online]. Available: http://arxiv.org/abs/1610.03771

[17] C. Sun, L. Huang, and X. Qiu, "Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence," *CoRR*, vol. abs/1903.09588, 2019. [Online]. Available: http://arxiv.org/abs/1903.09588

[18] G. Gennaro and E. Ash, "Emotion and reason in political language," *The Economic Journal*, vol. 132, no. 643, pp. 1037–1059, 2022.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[21] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[22] P. Schneider, T. Schopf, J. Vladika, M. Galkin, E. Simperl, and F. Matthes, "A Decade of Knowledge Graphs in Natural Language Processing: A Survey," in *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online only: Association for Computational Linguistics, Nov. 2022, pp. 601–614. [Online]. Available: https://aclanthology.org/2022.aacl-main.46

[23] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, "Relation Classification via Convolutional Deep Neural Network," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, Aug. 2014, pp. 2335–2344. [Online]. Available: https://aclanthology.org/C14-1220

[24] P.-L. Huguet Cabot and R. Navigli, "REBEL: Relation Extraction by End-to-end Language Generation," in *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 2370–2381. [Online]. Available: https://aclanthology.org/2021.findings-emnlp.204

[25] M. J. S. R. L. Z. Ledell Wu, Fabio Petroni, "Zero-shot Entity Linking with Dense Entity Retrieval," in *EMNLP*, 2020.

[26] E. Barba, L. Procopio, and R. Navigli, "ExtEnD: Extractive Entity Disambiguation," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Online and Dublin, Ireland: Association for Computational Linguistics, May 2022.

[27] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, and D. Lin, "Knowledge Base Completion via Search-Based Question Answering," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 515–526. [Online]. Available: https://doi.org/10.1145/2566486.2568032

[28] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[29] S. Dumancic, A. García-Durán, and M. Niepert, "A comparative study of distributional and symbolic paradigms for relational learning," *arXiv preprint arXiv:1806.11391*, 2018.

[30] H. Chen, S. F. Sultan, Y. Tian, M. Chen, and S. Skiena, "Fast and accurate network embeddings via very sparse random projection," in *Proceedings of the 28th ACM international conference on information and knowledge management*, 2019, pp. 399–408.

[31] L. Hu, T. Yang, L. Zhang, W. Zhong, D. Tang, C. Shi, N. Duan, and M. Zhou, "Compare to the knowledge: Graph neural fake news detection with external knowledge," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 754–763.

[32] E. Palumbo, G. Rizzo, R. Troncy, E. Baralis, M. Osella, and E. Ferro, "Knowledge graph embeddings with node2vec for item recommendation," in *The Semantic Web: ESWC 2018 Satellite Events: ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers 15*. Springer, 2018, pp. 117–120.

[33] L. Page, S. Brin, R. Motwani, and T. Winograd, "The Pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[34] M. Connelly, R. Hicks, R. Jervis, and A. Spirling, "New evidence and new methods for analyzing the Iranian revolution as an intelligence failure," *Intelligence and National Security*, vol. 36, no. 6, pp. 781–806, 2021.

[35] T. N. Archives, "The National Archives," Jul 2022. [Online]. Available: https://www.nationalarchives.gov.uk/webarchive/

[36] D. Beavan and F. Nanni, "Data Study Group Final Report: The National Archives, UK: Discovering Topics and Trends in the UK Government Web Archive," *[""]*, 2021.

[37] A. M. Dai, C. Olah, and Q. V. Le, "Document embedding with paragraph vectors," *arXiv preprint arXiv:1507.07998*, 2015.

[38] L. Rheault and C. Cochrane, "Word embeddings for the analysis of ideological placement in parliamentary corpora," *Political Analysis*, vol. 28, no. 1, pp. 112–133, 2020.

[39] K. Dritsa, A. Thoma, I. Pavlopoulos, and P. Louridas, "A Greek Parliament Proceedings Dataset for Computational Linguistics and Political Analysis," *Advances in Neural Information Processing Systems*, vol. 35, pp. 28 874–28 888, 2022.

[40] "Politics Ontology." [Online]. Available: https://iptc.org/thirdparty/bbc-ontologies/politics.html

[41] D. Schwabe, C. Laufer, and A. J. G. Busson, "Building Knowledge Graphs About Political Agents in the Age of Misinformation," *CoRR*, vol. abs/1901.11408, 2019. [Online]. Available: http://arxiv.org/abs/1901.11408

[42] B. Abu-Salih, M. Al-Tawil, I. Aljarah, H. Faris, and P. Wongthongtham, "Relational Learning Analysis of Social Politics using Knowledge Graph embedding," *CoRR*, vol. abs/2006.01626, 2020. [Online]. Available: https://arxiv.org/abs/2006.01626

[43] N. Stoehr, L. T. Hennigen, S. Ahbab, R. West, and R. Cotterell, "Classifying Dyads for Militarized Conflict Analysis," *arXiv preprint arXiv:2109.12860*, 2021.

[44] S. Feng, Z. Chen, W. Zhang, Q. Li, Q. Zheng, X. Chang, and M. Luo, "KGAP: Knowledge Graph Augmented Political Perspective Detection in News Media," *arXiv preprint arXiv:2108.03861*, 2021.

[45] https://github.com/datasets/world-cities.

[46] "GeoNames." [Online]. Available: www.geonames.org

[47] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength Natural Language Processing in Python," 2020.

[48] X. Cao and B. Liu, "CS 224N Final Project: Unsupervised Clustering of People, Places, and Organizations in Wikileaks Cables with NLP cues," 2011.

[49] https://github.com/sloria/TextBlob.

[50] "Neo4j Link Prediction Pipeline." [Online]. Available: https://neo4j.com/docs/graph-data-science/current/machine-learning/linkprediction-pipelines/link-prediction/

[51] "Importing CSV data into Neo4j." [Online]. Available: https://neo4j.com/docs/getting-started/data-import/csv-import/

[52] "Foreign Policy of the Jimmy Carter Administration." [Online]. Available: https://en.wikipedia.org/wiki/Foreign_policy_of_the_Jimmy_Carter_administration

Appendix A

Appendix

```
SELECT ?item WHERE {
SERVICE wikibase:mwapi {
    bd:serviceParam wikibase:endpoint "www.wikidata.org";
    wikibase:api "EntitySearch";
    mwapi:search  \'"""+name+"""\';
    mwapi:language "en".
    ?item wikibase:apiOutputItem mwapi:item.
    ?num wikibase:apiOrdinal true.
}
?item wdt:P31 wd:Q5
}
```

Figure A.1: SPARQL query for searching person with ’name’ in Wikidata

```
SELECT ?item ?itemLabel ?startyearLabel ?endyearLabel
WHERE  {
    wd:"""+Q+""" p:P102 ?statement1.
    ?statement1 ps:P102 ?item.
    OPTIONAL{?statement1 pq:P580 ?startyear.}
    OPTIONAL{?statement1 pq:P582 ?endyear.}
    SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
```

Figure A.2: SPARQL query for searching a person’s (Q) political party membership and its start and end dates. P102 is member of a political party, P580 is start year, and P582 is end year.

```
SELECT ?country ?countryLabel WHERE {
SERVICE wikibase:mwapi {
    bd:serviceParam wikibase:endpoint "www.wikidata.org";
    wikibase:api "EntitySearch";
    mwapi:search  \'"""+name+"""\';
    mwapi:language "en".
    ?city wikibase:apiOutputItem mwapi:item.
    ?num wikibase:apiOrdinal true.
}
?city wdt:P31 wd:Q5119.
?city wdt:P17 ?country.
SERVICE wikibase:label { bd:serviceParam wikibase:language "en".}
}
```

Figure A.3: SPARQL query for searching if a city with 'name' is a capital. If so, returns its country. Q5119 is capital city, P31 is instance of, P17 is country.

```
match (d:Document)-[:REDACTED]-(r:Redaction)
where r.type = 'dollaramount'
with d, count(r) as r_count
match (t:BertTopicWithEntities)-[:ABOUT]-(d)
return t.description, count(d) as doc_count, sum(r_count) as tot_count,
    sum(r_count)/count(d) as tot_count_per_doc
order by tot_count desc
limit 20
```

Figure A.4: A Cypher query for answering the following question: "Which topics have highest count of 'dollar amount' redaction ?"

Rank	Person	Roles Held	PageRank Importance
1	A. Henry Kissinger	1,3	49.88
2	M. Nixon Richard	2	40.07
3	Cyrus R. Vance	1,3	35.39
4	David Dean	1	32.08
5	Dulles Foster John	1	30.09
6	P. Rogers William	1	25.93
7	Brzezinski Zbigniew	3	24.72
8	D. Dwight Eisenhower	2	24.27
9	Carter Jimmy	2	23.04
10	B. Johnson Lyndon	2	22.62

Table A.1: Top 10 Most Important Roles, and Corresponding Countries in FRUS. Numbers in Roles Held indicate ranks in Table 5.7.

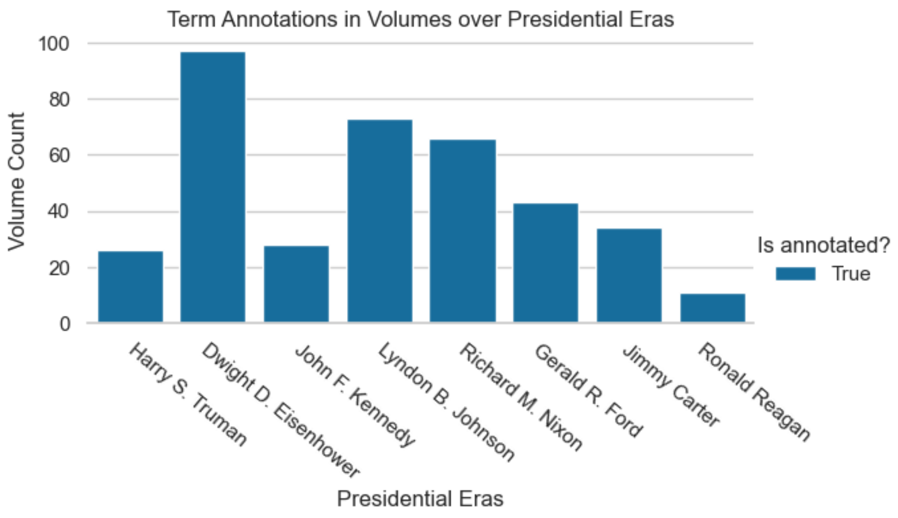
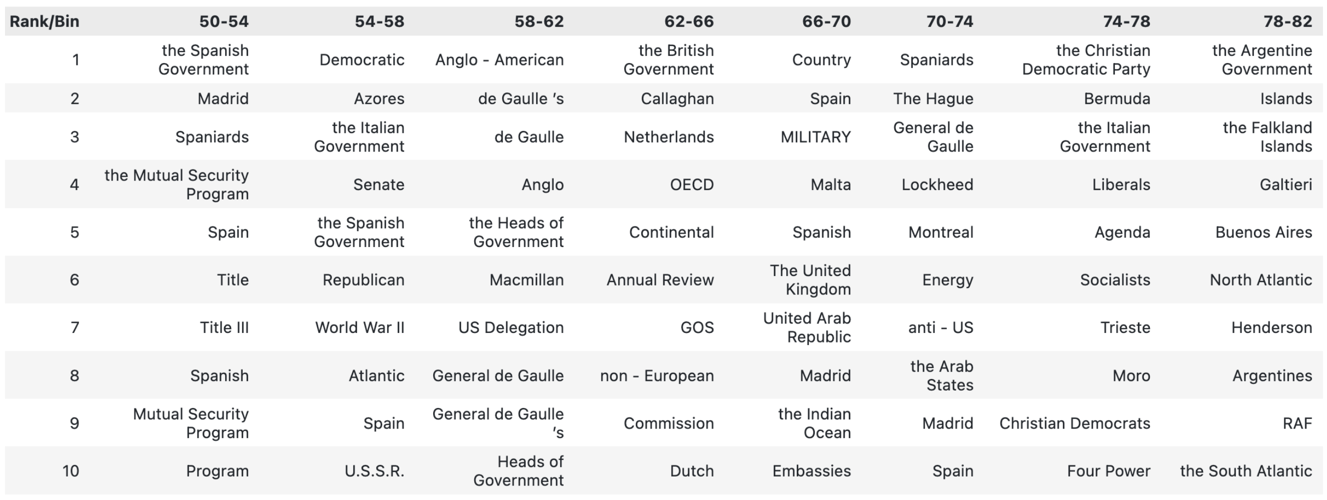


Figure A.5: Volumes with term annotations over presidential eras



Rank/Bin	50-54	54-58	58-62	62-66	66-70	70-74	74-78	78-82
1	the Spanish Government	Democratic	Anglo - American	the British Government	Country	Spaniards	the Christian Democratic Party	the Argentine Government
2	Madrid	Azores	de Gaulle 's	Callaghan	Spain	The Hague	Bermuda	Islands
3	Spaniards	the Italian Government	de Gaulle	Netherlands	MILITARY	General de Gaulle	the Italian Government	the Falkland Islands
4	the Mutual Security Program	Senate	Anglo	OECD	Malta	Lockheed	Liberals	Galtieri
5	Spain	the Spanish Government	the Heads of Government	Continental	Spanish	Montreal	Agenda	Buenos Aires
6	Title	Republican	Macmillan	Annual Review	The United Kingdom	Energy	Socialists	North Atlantic
7	Title III	World War II	US Delegation	GOS	United Arab Republic	anti - US	Trieste	Henderson
8	Spanish	Atlantic	General de Gaulle	non - European	Madrid	the Arab States	Moro	Argentines
9	Mutual Security Program	Spain	General de Gaulle 's	Commission	the Indian Ocean	Madrid	Christian Democrats	RAF
10	Program	U.S.S.R.	Heads of Government	Dutch	Embassies	Spain	Four Power	the South Atlantic

Figure A.6: Most Similar 10 Entities to Gibraltar over Years. Rank shows most similar to least. Bin shows the year span.

Annotation Type	Count
Person	2142
Term	3367

Table A.2: Annotated person and terms counts that are not mentioned in any volumes.

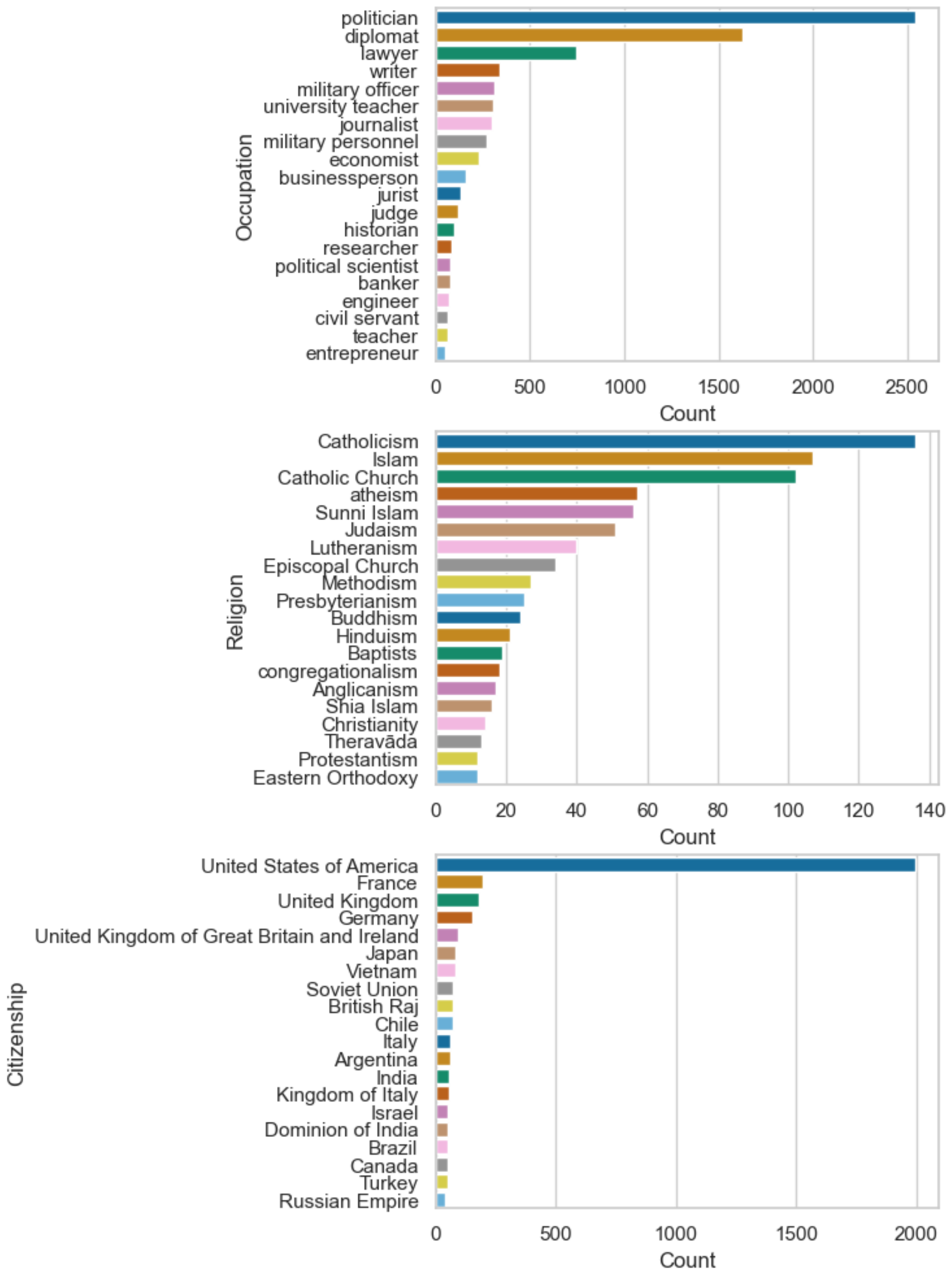


Figure A.7: Top 20 Most Frequent Occupations, Religions, and Citizenships in FRUS