# Total Relation Recall: High-Recall Relation Extraction

by

Xinyu Liu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2021

## Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

**Abstract**

As Knowledge Graphs (KGs) become important in a wide range of applications, including question-answering and recommender systems, more and more enterprises have recognized the value of constructing KGs with their own data. While enterprise data consists of structured and unstructured data, companies primarily focus on structured ones, which are easier to exploit than unstructured ones. However, most enterprise data are unstructured, including intranet, documents, and emails, where plenty of business insights live. Therefore, companies would like to utilize unstructured data as well, and KGs are an excellent way to collect and organize information from unstructured data.

In this thesis, we introduce a novel task, Total Relation Recall (TRR), that leverages the enterprise's unstructured documents to build KGs using high-recall relation extraction. Given a target relation and its relevant information, TRR aims to extract all instances of such relation from the given documents. We propose a Python-based system to address this task. To evaluate the effectiveness of our system, we conduct experiments on 12 different relations with two news article corpora. Moreover, we conduct an ablation study to investigate the impact of natural language processing (NLP) features.

**Acknowledgements**

First and foremost, I would like to thank my advisor, Professor Jimmy Lin, for his continuous guidance and support during the past three years since I knew him. It is still fresh in my memory that I started working with Jimmy as an undergraduate research assistant in the summer of 2018, which was the first time I got exposed to the fantastic world of research. It is a mystery how time flies and that I am about to graduate with my master's degree today. Not only did Jimmy motivate me to explore exciting technical problems, but he also constantly inspired me with his enthusiasm and optimism.

I would also like to thank the readers of my thesis, Professor Gordon Cormack and Professor Yaoliang Yu, for reviewing my work and providing invaluable comments.

Although I did not have the opportunity to spend lots of time in our Data Systems Group lab because of the COVID-19 pandemic, I am grateful to meet Jayden, Ralph, Brandon, Ryan, Reza, and the rest of the DSG members, who made my journey delightful and memorable.

I was really lucky to be accompanied by Tiffany and Jiaying in this challenging time of the pandemic. It would not be possible for me to find staying home enjoyable without their companionship and encouragement. Special thanks go to my fluffy cats Vinci and Yogurt, who have the superpower of always making me smile.

Finally, I would like to express my heartful gratitude to my family for their unwavering love, support, and understanding throughout my life.

## Dedication

This is dedicated to my parents for their unconditional love and support.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Knowledge Graphs (KGs), graph-based information representations focused on relations between concepts, have become a foundation for many enterprises to deliver powerful question-answering (QA) systems and make recommendations to their users. Although there is no formal definition of KG structures, they are typically constructed with nodes (representing entities such as organizations and people) and edges (representing relations such as *foundedBy*). A triple is defined as $\langle$*Subject, Relation, Object*$\rangle$, where *Subject* and *Object* are entities satisfying the relationship *Relation*. Figure 1.1 presents a KG subgraph consisting of two triples. The first triple $\langle$*Facebook, foundedBy, Mark Zuckerberg*$\rangle$ represents Facebook is founded by Mark Zuckerberg, and the second triple $\langle$*Mark Zuckerberg, yearOfBirth, 1984*$\rangle$ represents Mark Zuckerberg was born in 1984.


Figure 1.1: KG subgraph consisting of triples $\langle$*Facebook, foundedBy, Mark Zuckerberg*$\rangle$ and $\langle$*Mark Zuckerberg, yearOfBirth, 1984*$\rangle$.

Knowledge Graphs have attracted enterprises' attention because of their broad applications. For technology companies, KGs are considered a primary data source in building QA systems [6]. KG presented in Figure 1.1 could answer questions like "Who is the founder of Facebook?" or "What year was Mark Zuckerberg born?". Furthermore, media companies could enhance their recommendations' accuracy and diversity by integrating

**Sentence**: ***Facebook*** founder ***Mark Zuckerberg*** more than doubled his net worth, to $28.5 billion, and ranked 21st.[1]
**Relation**: ⟨*Facebook, foundedBy, Mark Zuckerberg*⟩

Figure 1.2: Relation *foundedBy* extracted from The Washington Post article.

**Sentence**: ***Mark Zuckerberg***, Facebook's chief executive, was not born until ***1984***.[2]
**Relation**: ⟨*Mark Zuckerberg, yearOfBirth, 1984*⟩

Figure 1.3: Relation *yearOfBirth* extracted from The Washington Post article.

KGs into their recommender systems [25]. For instance, if the user reads an article about *Mark Zuckerberg*, they might also be interested in articles related to *Facebook*. There are also KG-based applications in other domains, including medical, financial, and educational institutions [26].

In order to deliver robust applications to customers, enterprises aspire to build valuable and reliable KGs from their data source. Enterprise data are collected over years from various sources and stored in different formats, such as structured and unstructured data. Companies often concentrate on structured data, which is highly organized, explicitly defined, and usually stored in relational database management systems (RDBMSes). For example, if a company wanted to measure the success of its mobile app, it would collect information such as the number of daily active users and the amount of time users spend on it. However, lots of business intelligence can only be gained from unstructured data. As an illustration, by reading comments on Apple App Store or Google Play Store, the company would know which feature users enjoy the most and what improvements should be made in future. Unstructured data does not follow any predefined schema and makes up 80 percent of enterprise data [18]. Therefore, organizations should definitely leverage unstructured data to construct KGs, although it is more difficult to analyze than structured data.

Natural language processing (NLP) is a sub-discipline of artificial intelligence (AI), and it enables machines to read and understand human texts. Indeed, Relation Extraction

[1] https://www.washingtonpost.com/business/economy/national-business-and-economy-roundup/2014/03/03/f984abce-a31a-11e3-8466-d34c451760b9_story.html

[2] https://www.washingtonpost.com/opinions/how-income-inequality-benefits-everybody/2015/03/25/1122ee02-d255-11e4-a62f-ee745911a4ff_story.html

(RE) is an NLP task extracting well-structured relationships, typically between two or more entities, from unstructured texts. Once we have the extracted triple, adding its information to the KG is as simple as creating an edge in a graph. Figures 1.2 and 1.3 exhibit how we can extract triples $\langle$*Facebook, foundedBy, Mark Zuckerberg*$\rangle$ and $\langle$*Mark Zuckerberg, yearOfBirth, 1984*$\rangle$ from two sentences in the Washington Post articles. With this information, we could build the KG shown in Figure 1.1 by connecting two triples to their shared entity *Mark Zuckerberg*. When we want to find *all* instances of a particular relation, high recall becomes incredibly important. For example, if the enterprise needs to build a KG containing complete information about business founders, they would want to collect all triples of the *foundedBy* relation from the enterprise data.

In this thesis, we define a novel task, Total Relation Recall (TRR), helping an enterprise build KGs from its unstructured data. Let the relation triple be defined as $\langle S, R, O\rangle$ where $S, O$ are entities and $R$ is a relation, then a KG can be constructed by triples $\{\langle S_j, R_i, O_j\rangle\}_{j=1\ldots k}$. TRR assumes that the enterprise has an internal Knowledge Base (KB), and all entities in the enterprise data are correctly recognized and linked to the KB. Given this assumption, the TRR problem is defined as given a corpus of linked documents and a relation $R_i$ specified by the company, and we want to find all relation triples $\{\langle S_j, R_i, O_j\rangle\}_{j=1\ldots k}$, with each extracted triple being labelled by human assessors. The company can use these relation instances to either construct a new KG or enrich its existing KGs. For instance, given The Washington Post articles published in the past year with all their entities linked to Wikidata and a target relation *foundedBy*, our system aims to find all relation triples of format $\langle$*Organization, foundedBy, Person*$\rangle$ mentioned in those articles, where $\langle$*Facebook, foundedBy, Mark Zuckerberg*$\rangle$ might be one of them.

Moreover, we present a Python-based system providing a baseline solution to TRR, which is motivated by the Baseline Model Implementation (BMI) in the TREC Total Recall Track [17]. Our system is implemented based on the idea of continuous active learning (CAL) [3, 4] and works in an iterative style, where each iteration contains Retrieval, Classification, and Feedback Modules. Specifically, the Retrieval Module explores new relation candidates using relation keywords, and the Classification Module ranks all explored candidates and passes a batch of promising ones to be annotated by human assessors. After the assessment is completed, all annotated candidates are taken to the Feedback Module, which produces positive candidates to system output, revises the classifier used to rank candidates, and discovers unseen relation keywords that will be used to explore more candidates.

To evaluate our system's effectiveness, we take news articles from The Washington Post and The New York Times as a proxy for the enterprise data, and a commonly-used KB Wikidata is used as the enterprise's internal KB. As TRR assuming, all entities in articles

are recognized and linked to Wikidata items. In addition, we select 12 relations of four different entity types that are frequently mentioned in our life as target relations. The human assessment process is simulated with Wikidata that a candidate is labelled positive if and only if it can be verified on Wikidata. Evaluation results show that our proposed system can achieve reasonably high recall with feasible assessment efforts. Namely, by reading only 1.01% of the Washington Post corpus, our system is able to identify 88.45% of relation instances. In addition, we also conduct ablation studies on NLP features used to train the system's classifier, and the results show that the feature selection does not influence the effectiveness of our system much.

Although TRR and RE are both tasks of extracting relations from texts, they are different in that the latter is about generalization, while the former is about exhaustive enumeration. The RE system consists of the training phase and the generalization phase. In the training phase, the RE system is provided with a text corpus and all relation triples in the corpus, where triples are either labelled manually or generated from KBs automatically, and then the system will analyze NLP features associated with triples and learn to predict relation(s). Then we move to the generalization phase, in which we apply the trained system to extract triples from unseen datasets. Nevertheless, the goal of TRR is not to generalize the knowledge learned from the given dataset to unseen datasets. Instead, given a text corpus and a target relation, TRR only focuses on the given corpus and wants to extract all triples of the target relation from it. Another distinction between RE and TRR is that the RE system is static as it gets trained only once, but the TRR system is dynamic because it is adjusted over iterations. Hence, issues like overfitting exist in the RE system but not in the TRR system.

## 1.1 Contributions

The main contribution of this thesis can be summarized as follows:

- We formulate and introduce a novel task, Total Relation Recall, looking for all instances of a target relation by exhausting the given dataset. It is different from relation extraction, whose goal is to generalize the knowledge learned from one set of data to unseen data.
- We present an end-to-end system based on continuous active learning, which provides a baseline solution to the Total Relation Recall problem.

- In a simulation setup with Wikidata as human assessors, we evaluate our system with 12 relations of four different types over two collections of documents, and the experiment results show that our system is able to achieve reasonably high recall with practicable efforts. Moreover, we conduct an ablation study on NLP features used to train the classifier.

## 1.2 Thesis Organization

The thesis is organized as follows:

- Chapter 2 reviews a high-recall information retrieval task, TREC Total Recall Track, where the TRR problem is inspired from. Additionally, it introduces relation extraction, natural language processing, entity linking, and document indexing.
- Chapter 3 describes the TRR problem formulation, its CAL-based algorithm, and the implementation of a Python-based system in detail.
- Chapter 4 starts with experimental setups and evaluation metrics, followed by the primary experiment results on extracting 12 different relations over two news article corpora. Then we conduct the ablation study on NLP features and discuss our observations.
- Chapter 5 concludes this thesis and talks about future work.

# Chapter 2

# Background and Related Work

## 2.1 TREC Total Recall Track

### 2.1.1 Problem Definition

The Total Recall Track[1][2] is one of Text Retrieval Conference (TREC) tasks focusing on high-recall information retrieval [17, 16, 7]. This task takes a large corpus of documents and a topic description as input, with a goal of identifying all or substantially all relevant documents from the corpus using minimal annotation efforts. Besides, the task wants to find as many as possible relevant documents before non-relevant ones. The high-recall information retrieval can be applied to various real-world applications, such as the electronic discovery that collects *all* documents responsive to a legal request, the systematic review that looks for *all* published documents evaluating a method, and dataset construction that annotates *all* relevant documents from a corpus to build training/test set for future research.

[1] https://plg.uwaterloo.ca/~gvcormac/total-recall/
[2] https://plg.uwaterloo.ca/~gvcormac/total-recall/2016/guidelines.html

### 2.1.2 Algorithm and Implementation

In the Total Recall Track, a Baseline Model Implementation (BMI) is provided by coordinators to serve as a baseline solution to this task. The BMI is implemented based on an autonomous continuous active learning (CAL) system called AutoTAR [3, 4, 2]. In general, AutoTAR consists of a keyword search system and a learning algorithm. The keyword search part looks for a seed document and makes it our initial training set. The algorithm learns from the current training set and ranks documents in each batch by their likelihoods to be relevant. Human reviewers are given documents with the highest likelihood, one at a time, and they are immediately asked to use the best of their knowledge to evaluate and annotate these documents as "relevant" or "non-relevant" based on documents' relevance to the given topic. Then the annotated results are added to our current training set and will be used to further train the learning algorithm in the next batch. The process would terminate if a particular stopping criterion has been met; otherwise, it would move to the next batch. Algorithm 1 lists detailed steps in AutoTAR. The BMI is developed based on the AutoTAR approach, with the only modification being that the BMI chooses the Sofia ML[3] logistic regression as its classifier, while the AutoTAR uses SVMLight.[4]

**Algorithm 1:** The AutoTAR algorithm.

Step 1. Find a relevant "seed" document using ad-hoc search, or construct a synthetic relevant document from the topic description.
Step 2. The initial training set consists of the seed document identified in step 1, label "relevant".
Step 3. Set the initial batch size $B$ to 1.
Step 4. Temporarily augment the training set by adding 100 random documents from the collection, temporarily labelled "not relevant."
Step 5. Train an SVM classifier using the training set.
Step 6. Remove the random documents added in step 4.
Step 7. Select the highest-scoring $B$ documents for review.
Step 8. Review the documents, coding each as "relevant" or "not relevant."
Step 9. Add the documents to the training set.
Step 10. Increase $B$ by $\lceil \frac{B}{10} \rceil$.
Step 11. Repeat steps 4 through 10 until a sufficient number of relevant documents have been reviewed.

[3] https://code.google.com/archive/p/sofia-ml/
[4] https://www.cs.cornell.edu/people/tj/svm_light/

### 2.1.3 Experimental Setup

Eight test collections are used to evaluate the system's effectiveness. In each test collection, there are a corpus of documents and a given set of topics. For each topic, all documents have been labelled as "relevant" or "non-relevant". A sample test collection is 20 Newsgroups dataset,[5] comprising 18,828 documents across 20 different newsgroups. For this dataset, three newsgroups' categories are selected as topics for this collection: "space", "hockey", and "baseball". Another test collection is the Jeb Bush Emails dataset. This dataset contains 290,099 emails sent to Jeb Bush during his time as governor of Florida and ten topics including "new medical schools", "capital punishment", "manatee protection", etc. Furthermore, a Web server is used in experiments to simulate human reviewers, which contains all documents' relevance labels and can annotate any document in the corpus as "relevant" or "non-relevant" in real time. Hence, whenever a document is identified by the system, it will be immediately submitted to the Web server for relevance assessment rather than be presented to human reviewers, which automates the assessment process.

### 2.1.4 Evaluation Metrics

In the Total Recall Track, the effectiveness of systems is measured from multiple aspects. We introduce two of them here: the gain curve and the recall @ $aR + b$.

The gain curve describes the relationship between recall and assessment effort. For each point on the curve, its $x$-value represents the number of documents that have been annotated so far, and its $y$-value represents the proportion of relevant documents identified by the system to all relevant documents in the corpus. Figure 2.1 displays an example of gain curves, where each curve is generated from a different participant system. We can compare the effectiveness of systems by analyzing their curves vertically and horizontally. Let us first look at these curves vertically. Considering a fixed $x$-value, if the corresponding $y$-value on a system's gain curve is larger than the $y$-value on another curve, it means that the former system has found more relevant documents than the latter system after annotating $x$ documents. From this Figure, we can observe that after labelling 10,000 documents, the WaterlooClarke system has identified the most number of documents relevant to the specified topic. In the horizontal perspective, if we draw a horizontal line at the recall $y$ on the plot and look at each intersection point of the horizontal line and the gain curve, the smaller the point's $x$-value, the fewer assessment efforts required by the system to achieve recall $y$. Imagining there is a horizontal line at $y = 0.5$ in this Figure, then the

[5] http://qwone.com/~jason/20Newsgroups/

WaterlooCormack system's curve intersects with this line at the smallest $x$-value, which means the WaterlooCormack system needs to annotate the fewest documents to find 50% of all relevant documents.

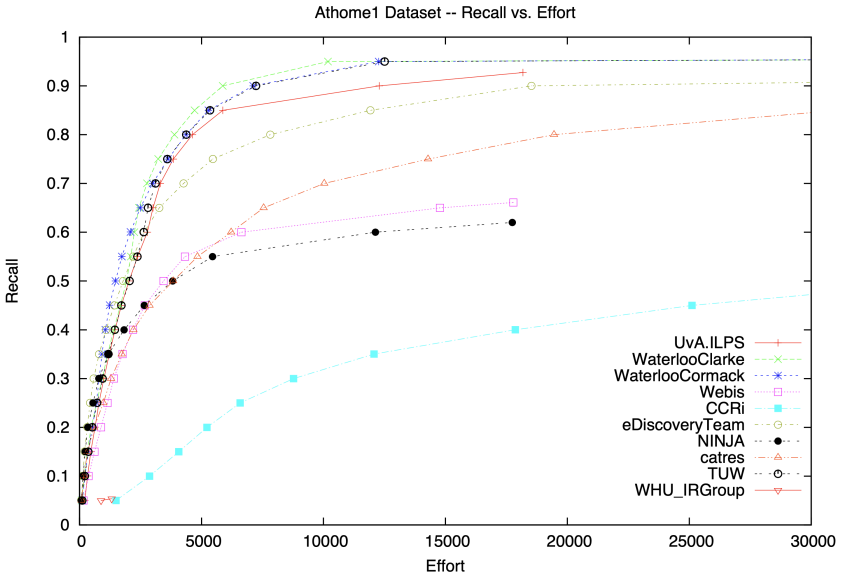


Figure 2.1: Gain curves in the TREC 2015 Total Recall Track results.

The recall @ $aR+b$ is defined as the recall achieved by a system after the Web server has assessed $aR+b$ documents, where $R$ is the number of all relevant documents in the corpus and $a, b$ can be any real number. This evaluation metric aims to measure the system's effectiveness at different stages of the process. For example, when $a = 1, b = 0$, the recall @ $aR + b$ becomes the recall @ $R$, representing the recall gained after $R$ documents are labelled. If we think in another way, the recall @ $R$ also tells us the current precision of these $R$ labelled documents. Since systems cannot guarantee 100% accuracy, documents identified by systems might actually be non-relevant to the specified topic, so the recall @ $R$ would be smaller than 100%. In order to find all or nearly all relevant documents, the Web server would have to annotate more than $R$ documents most times. We want to represent the number of annotated documents as a function of the number of all relevant documents, which is how the parametric form $aR + b$ is derived. Specifically, $a$ illustrates how many documents needed to be identified by the system to find each relevant document, and $b$ illustrates a fixed number of documents needed to be identified in addition to $aR$ documents to gain a better recall. Despite the specific value of $a$ and $b$, we always prefer a

larger value for recall @ $aR + b$ than a smaller one. For instance, let us say there are 1000 emails related to the "new medical schools" topic in the Jeb Bush Emails dataset, and we set $a = 2, b = 100$, then we will calculate the recall after the Web server annotates 2100 documents. If one system achieves a higher result than another system, we would know that more of the 2100 documents identified by the former system are relevant to "new medical schools" than by the latter system, leading to both better recall and precision.

### 2.1.5 Results

In both 2015 and 2016 Total Recall Tracks, BMI has accomplished superior results [17, 7]. There are 15 systems submitted by participants in 2015 and 2016. Some systems have achieved slightly better effectiveness in specific test collections, but none of them has consistently beaten the BMI. An interesting observation is that systems achieving better effectiveness always take much longer run-time than the BMI.

## 2.2 Relation Extraction

Relation extraction is a task of extracting relations from texts. The relation could be written as triples $\langle S,\ R,\ O \rangle$, where $S$ and $O$ are called entities, and $R$ is the relation name. For example, the sentence

> "Back in 2014, ***Microsoft*** co-founder ***Paul Allen*** launched the Allen Institute for Artificial Intelligence, which uses the tagline 'AI for the common good.'"[6]

contains the information that *Paul Allen* is the founder of *Microsoft*, which can be represented as a triple $\langle$*Microsoft, foundedBy, Paul Allen*$\rangle$. We can also say that the entity pair $\langle$*Microsoft, Paul Allen*$\rangle$ is an instance of relation *foundedBy*, or the entity pair satisfies the *foundedBy* relation.

In previous work, relation extraction has been done through supervised [14, 13], unsupervised [19, 5], and distantly supervised approaches [12, 15, 10, 20]. All of them have some limitations. The supervised method is too expensive since it requires huge human effort to construct the training set. For the unsupervised method, although it has a relatively low cost, its results are generally suboptimal. The distant supervision is a paradigm

[6] https://www.washingtonpost.com/news/the-switch/wp/2015/12/14/tech-titans-like-elon-musk-are-spending-1-billion-to-save-you-from-terminators/

proposed for automated relation extraction, which uses existing KBs as the training set source. However, a drawback is that distant supervision only works for relations contained in KBs. Many interesting relations in real life are not covered in KBs, such as *favoriteVacationPlace*.

## 2.3 Natural Language Processing

This section introduces four NLP features helping machines understand raw texts: tokenization, part-of-speech tagging, dependency parse, and named entities.

The tokenization process takes raw texts as input and divides them into tokens, such as words and punctuation. Table 2.1 shows the tokenization result of the sentence "Last night I played my guitar loudly and the neighbors complained.", where white spaces split tokens.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Last | night | I | played | my | guitar | loudly | and | the | neighbours | complained | . |

Table 2.1: Tokenization output.

The Part-of-speech tagger (POS tagger) attributes part-of-speech categories to tokens generated from the tokenization step. There are many kinds of POS tags, we use the Universal POS (UPOS) tags listed in Table 2.2[7] as an example. Given a sentence "Last night I played my guitar loudly and the neighbors complained.", Table 2.3 indicates which UPOS category each token falls into.

A named entity can be seen as an item in pre-defined categories, including *Person*, *Organization*, or *Location*. Given a sentence, the named entity recognizer (NER) tries to identify entities appearing in it. For example, as shown in Figure 2.2,[8] given a sentence "Anita Brookner was born July 16, 1928, in London.",[9] the NER recognizes that *Anita Brookner* is a *Person* entity, *July 16, 1928* is a *Date* entity, and *London* is a *Location* entity.

[7] https://universaldependencies.org/docs/u/pos/

[8] https://explosion.ai/demos/displacy-ent

[9] https://www.washingtonpost.com/entertainment/books/anita-brookner-booker-prize-winning-author-of-ruminative-novels-dies-at-87/2016/03/15/c6601012-eac6-11e5-a6f3-21ccdbc5f74e_story.html

| Open Class Words | Closed Class Words | Other |
|---|---|---|
| ADJ: adjective | ADP: adposition | PUNCT: punctuation |
| ADV: adverb | AUX: auxiliary verb | SYM: synbol |
| INTJ: interhection | CONJ: coordinating conjunction | X: other |
| NOUN: noun | DET: determiner | |
| PROPN: proper noun | NUM: numeral | |
| VERB: verb | PART: particle | |
| | PRON: pronoun | |
| | SCONJ: subordinating conjunction | |

Table 2.2: Universal POS tags.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Last | night | I | played | my | guitar | loudly | and | the | neighbours | complained | . |
| ADJ | NOUN | PRON | VERB | DET | NOUN | ADV | CONJ | DET | NOUN | VERB | PUNCT |

Table 2.3: POS tagger output.


Figure 2.2: Named entities recognizer output.

The dependency parser examines texts' syntactic structures, where arcs connect head words to their child words with arc labels being grammatical relationships between two words. Figure 2.3[10] shows the dependency path for the sentence "She arrived early for the meeting.". The arc connecting word *arrived* and *early* represents that *early* describes *arrived* with the syntactic relationship *advmod*.

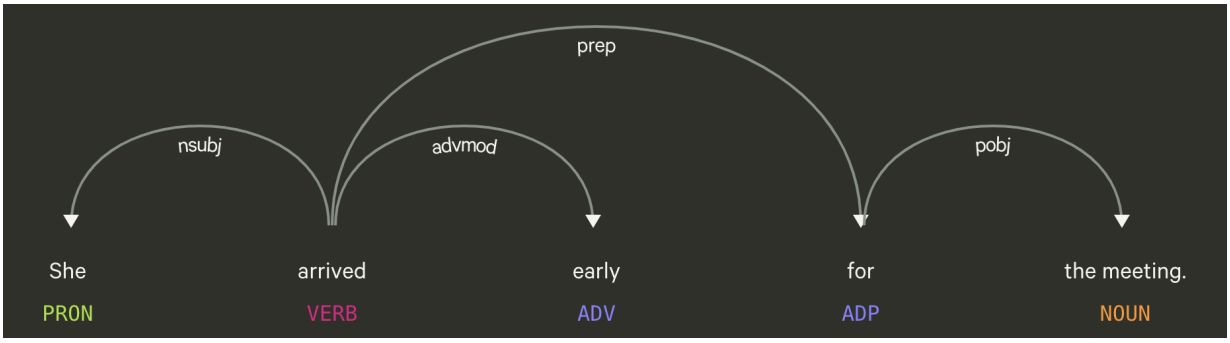


Figure 2.3: Dependency parser visualization.

spaCy is an open-source software library offering a variety of NLP tools [8], including all linguistic features mentioned above. Though there are many other well-known NLP libraries such as CoreNLP [11] and NLTK [1], spaCy is preferred when people want to develop an end-to-end production system, which should be adequate on CPU. A pre-trained multi-task Convolutional Neural Network (CNN) model called *en_core_wb_lg* is provided by spaCy, equipping us with an NLP pipeline consisting of a tokenizer, a tagger, a dependency parser and an entity recognizer. As presented in Figure 2.4,[11] this pipeline transforms texts to spaCy *Doc* objects that store all relevant NLP features obtained from the texts. spaCy also offers a serialization method that saves its *Doc* object as a binary file so that it can be restored very fast later when we need it.

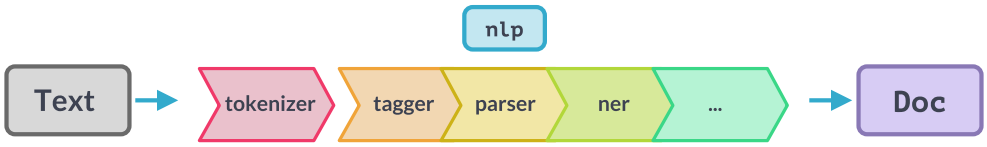


Figure 2.4: spaCy NLP pipeline.

[10] https://explosion.ai/demos/displacy
[11] https://spacy.io/pipeline-fde48da9b43661abcdf62ab70a546d71.svg

## 2.4 Entity Linking

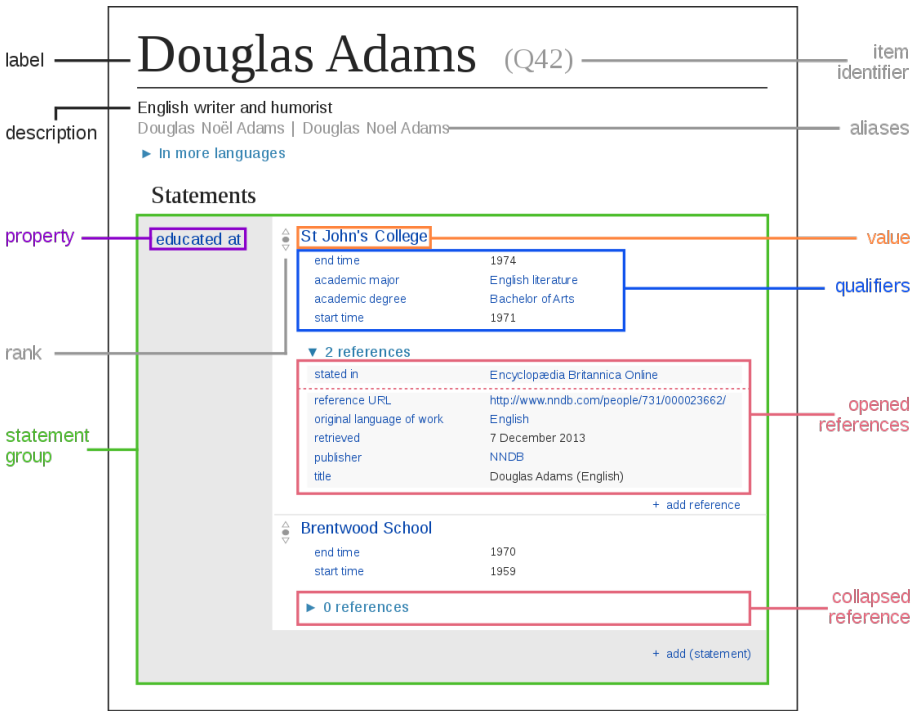


Figure 2.5: Wikidata item.

Knowledge Base (KB), such as Wikipedia or Freebase, gathers detailed structured and unstructured information about the world. Wikidata is a KB collecting Wikipedia data together and presents them in a more structured format [22]. As of January 2021, there are 92,678,133 items in Wikidata, including human, organization, etc. Figure 2.5[12] exhibits an example of Wikidata item *Douglas Adams*, and let us concentrate on its four fields: label, item identifier, property and value. The label is the name of the item, which may not be unique. For example, there are two items with the same label *Nike*, where one of them is an American athletic equipment company,[13] while another one is the goddess of victory in Greek mythology.[14] To distinguish items with the same label, Wikidata requires a unique identifier (also named "QID") for each item, made of the upper-case letter "Q" followed by a positive integer. In the previous example, items with QID *Q483915* and *Q165023*

[12] https://en.wikipedia.org/wiki/Wikidata
[13] https://www.wikidata.org/wiki/Q483915
[14] https://www.wikidata.org/wiki/Q165023

represent the American company and the victory goddess respectively. The property is related to our relation name, and the value is an entity in our relation mention. For example, consider the property *educated at* and the value *St John's College* in Figure 2.5, they tell us that Douglas Adams is educated at St John's College. In other words, the entity pair ⟨*Douglas Adams, St John's College*⟩ is an instance of the relation *educated at*.

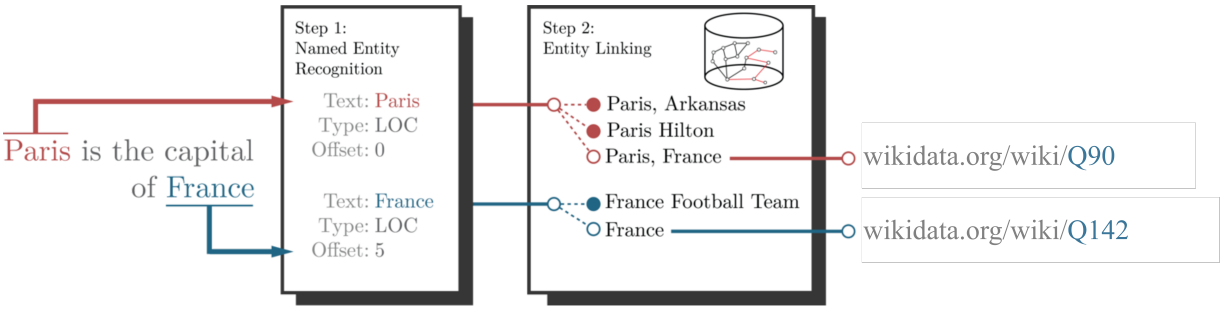


Figure 2.6: Entity linking process.

Given raw texts and their NER results, the entity linking process matches the recognized entities to their corresponding KB items. For example, given the sentence "Paris is the capital of France", we want to link entities occurring in this sentence to Wikidata items. After completing the NER step, *Paris* and *France* are recognized as *Location* entities in Figure 2.6.[15] However, it is not straightforward to link them to Wikidata items because there are more than one items related to labels *Paris* and *France* in the KB. The entity disambiguation is a critical component of the entity linking task, investigating potential Wikidata candidates for each entity and selecting the most assuring one. In our example, the most assuring candidates are *Paris (Q90)*[16] that is the capital and largest city of France, and *France (Q142)*[17] that is the country in Western Europe. Radboud Entity Linker (REL) is a convenient entity linking tool [21], which provides both a Python package and a web API, and it supports switching between different KBs or NER tools.

[15] https://en.wikipedia.org/wiki/Entity_linking#/media/File:Entity_Linking_-_Example_of_pipeline.png
[16] https://www.wikidata.org/wiki/Q90
[17] https://www.wikidata.org/wiki/Q142

## 2.5 Document Indexing

Document indexing enables us to perform faster search and retrieval when there is a large corpus of documents. Pyserini is an information retrieval (IR) toolkit [9] providing the Python interface to Anserni IR toolkit [23, 24]. It is worth remarking on two Pyserini features. The first one is the sparse retrieval using bag-of-words representation, where user-defined documents are also supported. In this way, we could save texts with their NLP features as JSON documents, then index these documents for future access. For instance, the JSON document presented in Figure 2.7 contains the NER information of the sentence "The Manhattan Project and its atomic bomb helped bring an end to World War II.". Furthermore, Pyserini provides access to document vectors and raw term statistics, helping us obtain document-level features such as the TF-IDF score.

```
{
  "id": "doc1",
  "contents": "The Manhattan Project and its atomic bomb helped bring an end
               to World War II.",
  "NER": {
            "ORG": ["The Manhattan Project"],
            "EVENT": ["World War II"]
          }
}
```

Figure 2.7: JSON document supported by Pyserini.

# Chapter 3

# Total Relation Recall

## 3.1 Problem Definition

Nowadays, enterprises are familiar with data stored in RDBMS, and they make business decisions based on those structured data. For example, companies make a production plan for the next year based on their products' sales volumes in the current year. However, lots of enterprise data are not well-structured to be stored in RDBMS, such as articles, social media posts, and emails, but their information can be precious as well. To illustrate, if many fashion blogs said that yellow would be the most stylish colour in the next season, clothing companies would like to design and manufacture more yellow clothes even if yellow is not the best-selling colour currently. Hence, unstructured data plays an essential role in providing business intelligence. To help companies better utilize their unstructured data, we introduce a novel task, Total Relation Recall (TRR), that constructs KGs from a collection of documents.

The Total Relation Recall (TRR) problem is inspired by the TREC Total Recall Track, and both of them focus on the high-recall retrieval. While the Total Recall task is looking for *all* documents relevant to the given topic, TRR is looking for *all* facts of the target relation. Figure 3.1 displays the overview of the TRR problem. TRR assumes that the enterprise has its internal entity categories and KBs. Furthermore, all entities in its data are recognized with their entity categories and then correctly linked to KB items. Based on the assumption, TRR takes a collection of linked documents as the dataset and asks the enterprise to specify a target relation. Then it wants to find all facts of the target relation from the dataset. During the process of looking for relation facts, human assessors are involved in assessing each candidate fact's correctness, but the human efforts should be

limited to reduce the cost of labour. Specifically, TRR wants to use as few human efforts as possible to find nearly all entity pairs expressing the target relation in the given document collection. We will discuss the task assumption, input, human assessment and output in detail.

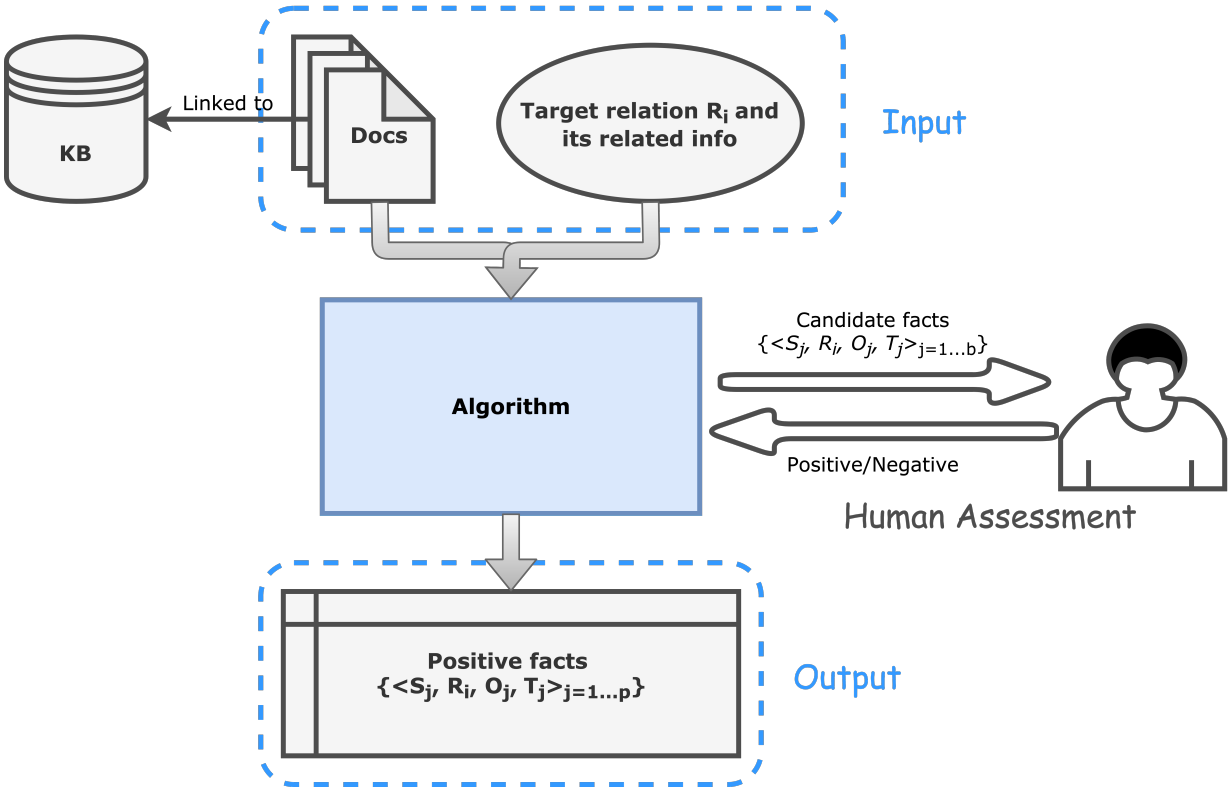


Figure 3.1: Overview of TRR problem.

### 3.1.1 Assumptions

In the task assumption, the enterprise has its internal entity categories and KBs, and every entity that appeared in the enterprise data has been recognized with its entity category and linked to the correct item in KBs. An enterprise can have various internal entity types, such as a *Person* category containing all employees or a *Location* category including all offices' addresses. Similarly, internal KBs can be in many different formats. For example, the human resource system storing employee information can be considered as a KB, where each KB item contains the information about an employee, including a unique

employee ID, his/her job title, team name, etc. Having all entities in the enterprise data recognized and linked to KBs means that if an entity mentioned in enterprise data has a corresponding item in the KB, it would be identified with its category, and there would be a link directly pointing the entity from the enterprise data to its KB item. There are many benefits of the NER and entity linking, and we will illustrate two of them here. The first benefit is disambiguation, which helps the enterprise distinguish entities with the same name. There can be more than one people named "John Smith" in a large enterprise, who may or may not have the same job title or work in the same team. If whenever "John Smith" is mentioned, it is consistently recognized as a *People* entity and linked to the correct employee page in the human resource system, then the enterprise would be able to distinguish between different people named "John Smith" by their unique employee IDs easily. The other benefit is deduplication, helping the enterprise to recognize the same entity even if they are mentioned by distinct aliases in the document. We consider the products' information system as an internal KB, where KB items contain specifications of different *Product* entities. Thinking about Apple Mac operating systems (macOSes), major macOSes get their own names other than version codes, like macOS version 11.2.3 is also called macOS Big Sur. Let us assume the enterprise has specified every mention of macOSes as a *Product* entity and linked it to the correct product specification in the information system. When a document talks about macOS version 11.2.3 and macOS Big Sur, it would be easy for the enterprise to recognize them as the same product because their links refer to the same KB item.

### 3.1.2 Input

There are two components in the task input: the linked enterprise data and the user-specified relation.

#### Dataset

In TRR, the dataset is a collection of linked documents provided by the enterprise, where documents can be emails received from the customer support team or news articles about company products. Based on our assumption, the enterprise has specified categories of all entities mentioned in the documents and linked them to corresponding KB items. Each entity is written as *(M, C, ID)*, where $M$ is the mention of the entity in enterprise data, $C$ is the entity's category, and $ID$ is the unique ID of the entity's linked KB item. To illustrate, given a sentence "I really appreciate Mr.Smith's help for fixing my laptop." from a customer email, the enterprise would generate a linked entity *(Mr.Smith, Person,*

*E000912)* associated to this sentence, where *Mr.Smith* is how the entity appears in enterprise documents, *Person* is the entity type, and *E000912* is the employee ID of the corresponding person in the enterprise's human resource system. TRR takes enterprise documents together with all their linked entities as the input dataset.

### Relation

In TRR, we use the KB IDs rather than entity names to represent entities with the purpose of disambiguation and deduplication mentioned in Section 3.1.1. For instance, the triple $\langle$*O000123, foundedBy, E000321*$\rangle$, where *O000123* and *E000321* are IDs referring to *Facebook* and *Mark Zuckerberg* in the enterprise KB, tells us that Facebook was founded by Mark Zuckerberg. For ease of understanding, we assume that enterprise KB IDs in the rest of this thesis are made by entity names so that the above relation triple can be written as $\langle$*Facebook, foundedBy, Mark Zuckerberg*$\rangle$.

As the second component of TRR input, the enterprise is expected to provide the following information to define a target relation:

- Relation name (e.g. *foundedBy*)
- Entity $S$ type (e.g. *ORG*)
- Entity $O$ type (e.g. *PERSON*)
- Relation keywords (e.g. founder, founded)
- Positive instances (e.g. $\langle$*Microsoft, Bill Gates*$\rangle$: "Bill Gates is a co-founder of Microsoft.")
- Single-valued vs. multi-valued
- Symmetric vs. asymmetric

First of all, the enterprise is asked to specify a target relation $R_i$ by defining the relation name and types of entities $S$ and $O$. There are many pre-defined categories for entities, such as *Person*, *Organization* or *Location*. Normally, each relation works for a specific type of entity pairs. For example, the *foundedBy* relation in the last example requires $S$ being an *Organization* entity and $O$ being a *Person* entity. In addition, the enterprise is also required to provide some relation keywords, which are words frequently used to describe the target relation. For example, "founder" is commonly used to describe the relation *foundedBy*.

Besides, TRR needs a small set of positive instances of the target relation (i.e., 5–15 instances), where each instance contains an entity pair $\langle S_j, O_j \rangle$ and a sentence $T_j$ that the entity pair comes from. The instance is positive if a human were able to learn that $\langle S_j, O_j \rangle$ satisfies $R_i$ after reading its sentence $T_j$. As an illustration, considering an instance with an entity pair $\langle$*Microsoft, Bill Gates*$\rangle$ and a sentence "Bill Gates is a co-founder of Microsoft.", because it is evident that the given entity pair satisfies the target relation *foundedBy* by reading the sentence, we would consider this instance as positive. However, given another instance with the same entity pair $\langle$*Microsoft, Bill Gates*$\rangle$ but a different sentence "Bill Gates has left Microsoft's board.", they cannot be counted towards positive instances since the given sentence does not provide enough information to determine whether the relation *foundedBy* exists between *Microsoft* and *Bill Gates*.

Lastly, the enterprise is asked to specify two properties of the target relation. The first property is whether the relation is single-valued or multi-valued. A relation $R_i$ is single-valued if each entity $S_j$ has exactly one entity $O_j$ respect to $R_i$, for example, the relation *yearOfBirth* is single-valued because everyone has a unique birth year. If an entity $S_j$ can form the relation instance with different entities, it is a multi-valued relation. For example, the relation *foundedBy* is multi-valued because an organization can be co-founded by more than one person. The second property is symmetry. For a symmetric relation $R_i$, if an entity pair $\langle S_j, O_j \rangle$ is an instance of $R_i$, then another relation instance can be built by interchanging two entities $\langle O_j, S_j \rangle$. Entities are not interchangeable in instances of asymmetric relations. To illustrate, the *sibling* relation is symmetric since entity $S, O$ are siblings to each other, yet the *father* relation is asymmetric since entity $S_j$'s father is entity $O_j$ but not the other way around.

### 3.1.3 Human Assessment

After the corpus of documents and the enterprise-specified relation are ready, TRR will put them into an algorithm whose goal is to find all entity pairs expressing the target relation in the given corpus. Although there are no strict restrictions on the algorithm implementation or techniques used, human assessors are involved in the algorithm to review and label extracted relation facts. The labelling standard is the same as what is used to generate positive instances in the training set. Concretely, algorithm works on extracting candidate facts $\langle S_j, R_i, O_j, T_j \rangle$ with $R_i$ being the target relation, $S_j, O_j$ being entities and $T_j$ being the sentence where $S_j, O_j$ are from. These candidate facts will be presented to human assessors immediately after they are extracted by the algorithm. Then assessors are asked to label each fact as "positive" or "negative" depending on whether the sentence $T_j$ proves that the target relation $R_i$ exists between entities $S_j$ and $O_j$. After the assessment is

completed, the labelling results will be taken back to the algorithm. TRR assumes human reviewers always use the best of their knowledge in labelling candidate facts.

### 3.1.4 Output

The output of the TRR task is all facts $\{\langle S_j, R_i, O_j, T_j \rangle_{j=1...p}\}$ that are labelled positive by human assessors. Considering a TRR task with the target relation $R_i$ being *foundedBy* and the dataset being all articles from past Forbes Magazines, then $\langle$*Microsoft, foundedBy, Bill Gates, "Bill Gates is a co-founder of Microsoft."*$\rangle$ and $\langle$*Facebook, foundedBy, Mark Zuckerberg, "Facebook was founded by Mark Zuckerberg and fellow Harvard University students."*$\rangle$ are possible output facts.

After positive facts $\{\langle S_j, R_i, O_j, T_j \rangle_{j=1...p}\}$ are produced from TRR, they can be used to create a new KG or expand existing KGs. For example, if there are four facts in the TRR's output (we omit $T_j$ here for simplicity): $\langle$*Microsoft, foundedBy, Bill Gates, ...*$\rangle$, $\langle$*Microsoft, foundedBy, Paul G. Allen, ...*$\rangle$, $\langle$*Allen Institute, foundedBy, Paul G. Allen, ...*$\rangle$ and $\langle$*Allen Institute, foundedBy, Jody Allen, ...*$\rangle$, then a new KG shown in Figure 3.2 can be established using the output.

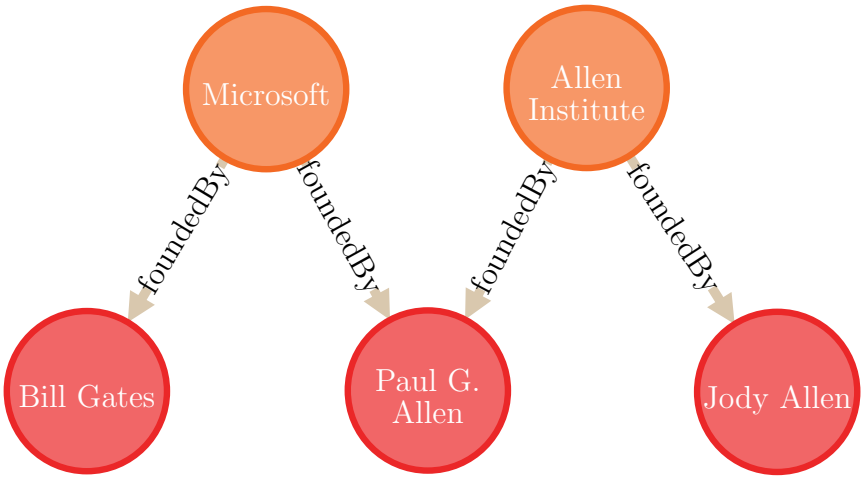


Figure 3.2: KG constructed from TRR output facts.

## 3.2 Algorithm

As discussed in Section 2.1.2, a system called BMI based on continuous active learning (CAL) has achieved superior results in the TREC Total Recall Track. Though the Total Recall task looks for relevant documents and our TRR problem extracts relevant relation facts, they are similar in that both tasks aim to find nearly all the relevant items from the given corpus, with human assessments involved. Hence, it makes sense to work on our TRR problem using CAL.

This section introduces a CAL-based algorithm working toward the TRR task, which corresponds to the blue box in Figure 3.1. The overview of our CAL-based algorithm is drawn in Figure 3.3 and also listed in Algorithm 2. The algorithm takes the TRR input, which is a corpus of documents with entities recognized and linked to enterprise KBs and a target relation $R_i$ with its related information specified by the enterprise, and it wants to extract as many entity pairs $\langle S_j, O_j \rangle$ satisfying the relation $R_i$ as possible from the given corpus. Our algorithm works in an iterative style, where each iteration consists of Retrieval, Classification, and Feedback Modules.

As the first step of each iteration, the Retrieval Module looks for candidate entity pairs with contexts from the given corpus using relation keywords, where a candidate entity pair with context is defined as $\langle S_j, O_j, T_j \rangle$ with $\langle S_j, O_j \rangle$ being the entity pair, and $T_j$ being the sentence that entities $S_j, O_j$ come from. Then these discovered candidates are added into an unlabelled candidate set containing all unlabelled candidate entity pairs with contexts. The next step is to feed all candidates in the unlabelled candidate set into the Classification Module to be ranked based on their likelihoods of expressing the target relation. In each iteration, human assessors are given a batch of $b$ candidate facts to review and label, so our Classification Module looks for the most promising $b$ candidate entity pairs with contexts $\{\langle S_j, O_j, T_j \rangle\}_{j=1...b}$ and pass them to human assessors as a batch of candidate facts $\{\langle S_j, R_i, O_j, T_j \rangle_{j=1...b}\}$. After the human assessment is completed, the assessment results are taken to our Feedback Module, which outputs all positive facts, explores new relation keywords for the Retrieval Module, and further improves the classifier used in the Classification Module. If there is no new relation keyword explored in the current iteration, our algorithm will skip the Retrieval Module and simply apply the revised classifier to the existing unlabelled candidate set in the next iteration. At the end of each iteration, the enterprise is asked whether they would like to continue to the next iteration or terminate the algorithm. When the enterprise thought enough positive facts had been identified, they would terminate the algorithm and construct KGs using all facts produced by the algorithm. In the rest of this section, we will discuss how each module works in detail.

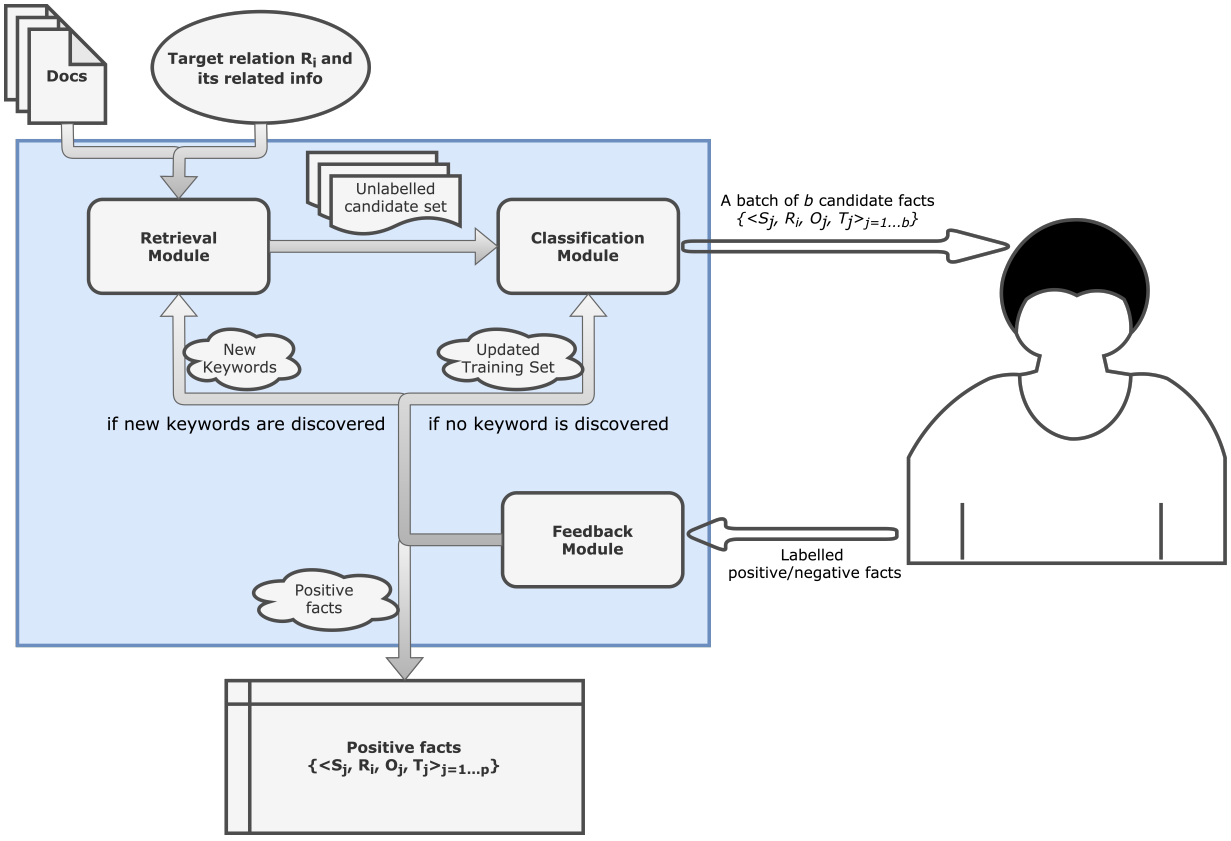


Figure 3.3: Algorithm Overview.

### 3.2.1 Retrieval

We say a candidate entity pair with context $\langle S_j, O_j, T_j \rangle$ satisfying the target relation $R_i$ if the target relation $R_i$ can be observed between entities $S_j, O_j$ by reading the sentence $T_j$. When the enterprise defines $R_i$, they also specify types of entities associated with $R_i$. Hence, a candidate has the probability of satisfying the target relation only if its entities $S_j, O_j$ match the required entity types. In other words, if its entities are not in the same type as the enterprise specified, this candidate must not express the target relation. For example, suppose the enterprise defines a relation $foundedBy$ with entity $S$ being *Organization* and entity $O$ being *Person*. In that case, we know that any candidate $\langle S_j, O_j, T_j \rangle$ with $S_j, O_j$ being other entity types, such as $\langle$*Microsoft, Facebook, ...*$\rangle$, cannot be an instance of the target relation without needed to read its sentence $T_j$. For a candidate $\langle S_j, O_j, T_j \rangle$ with the same entity types as specified, it may or may not be a relation

**Algorithm 2:** CAL-based TRR algorithm.

Step 1. If there are any unused relation keywords, use these keywords in the Retrieval Module to search for valid candidate entity pairs with contexts $\{\langle S_j, O_j, T_j\rangle\}_{j=1...m}$, add them to the unlabelled candidate set, and mark these keywords as "used"; otherwise, skip this step;

Step 2. Run Classification Module to find the $b$ most promising candidate entity pairs with contexts $\{\langle S_j, O_j, T_j\rangle\}_{j=1...b}$ from the unlabelled candidate set and transform them to $b$ candidate facts $\{\langle S_j, R_i, O_j, T_j\rangle\}_{j=1...b}$;

Step 3. Pass a batch of $b$ candidate facts produced in Step 2 to human reviewers for assessments, where each fact will be labelled as "positive" or "negative";

Step 4. Take assessment results to Feedback Module, which outputs positive facts, explores new relation keywords, and updates the training set;

Step 5. Repeat steps 1 through 4 until the enterprise asks to terminate the algorithm.

instance. Thinking about two examples ⟨*Microsoft, Bill Gates, "Bill Gates is a co-founder of Microsoft."*⟩ and ⟨*Microsoft, Mark Zuckerberg, "Mark Zuckerberg visited Microsoft campus yesterday."*⟩, both of them satisfy the entity type requirement. Whereas, only the first candidate is an instance of relation *foundedBy*, which requires the human assessor to read the sentence $T_j$ to determine. In our algorithm, we only care about candidates $\langle S_j, O_j, T_j\rangle$ with the specified entity types, called "valid" candidate entity pairs with contexts in the rest of this thesis.

Our Retrieval Module's goal is to search for valid candidate entity pairs with contexts that likely express the target relation $R_i$ from the document corpus. In theory, all valid candidates $\langle S_j, O_j, T_j\rangle$ have probabilities of satisfying the target relation, but there can be tons of such candidates in the corpus. Our module would like to find those candidates with relatively high probabilities of expressing the target relation. As we know, a candidate $\langle S_j, O_j, T_j\rangle$ is considered as an instance of the target relation $R_i$ if and only if its sentence $T_j$ proves the relation $R_i$ exists between entities $S_j, O_j$, so a candidate $\langle S_1, O_1, T_1\rangle$ would have higher chance to be the target relation's instance if its sentence $T_1$ is relevant to $R_i$ than another candidate $\langle S_2, O_2, T_2\rangle$ whose sentence $T_2$ is completely irrelevant to $R_i$. Furthermore, we could use relation keywords to determine whether a sentence $T_j$ is relevant to the relation $R_i$ by checking if $T_j$ mentions any relation keywords. The relation keywords are first populated by enterprise-provided input (see Section 3.1.2), and then we explore more keywords in the Feedback Module (see Section 3.2.3).

In specific, our Retrieval Module wants to collect all valid candidate entity pairs with

contexts $\{\langle S_j, O_j, T_j\rangle\}_{j=1...k}$ whose sentence $T_j$ is related to at least one relation keyword and add them into an unlabelled candidate set. The Retrieval Module works in two steps: sentence retrieval and candidate generation. The sentence retrieval step looks for all sentences relevant to at least one relation keyword from the given documents. Assuming the enterprise has defined a relation *foundedBy* with relation keywords {*founded, co-founder*}, and there are two sentences in a document: "Bill Gates is a co-founder of Microsoft." and "Mark Zuckerberg visited Microsoft campus yesterday.", then only the former sentence would be retrieved in this step because it contains the keyword *co-founder*. Next, the candidate generation part works on looking for all valid candidate entity pairs with contexts from retrieved sentences. Concretely, assuming the target relation $R_i$ requires two entities of categories $A$ and $B$, then for any sentences retrieved from the last step containing at least one $A$ entity and one $B$ entity, we would be able to generate valid candidate(s) from it. When there are more than one $A$ or $B$ entities in a sentence, there could be multiple valid candidates identified from this sentence. For example, given a target relation *foundedBy* requiring an *Organization* entity and a *Person* entity a sentence "The Washington Post is owned by Amazon founder and chief executive Jeffrey P. Bezos."[1] with two *Organization* entities {*The Washington Post, Amazon*} and one *Person* entity *Jeffrey P. Bezos*, we can generate two valid candidates from it: ⟨*The Washington Post, Jeffrey P. Bezos, "The Washington Post is owned by Amazon founder and chief executive Jeffrey P. Bezos."*⟩ and ⟨*Amazon, Jeffrey P. Bezos, "The Washington Post is owned by Amazon founder and chief executive Jeffrey P. Bezos."*⟩. Our algorithm maintains an unlabelled candidate set storing all unlabelled candidate entity pairs with contexts. Whenever the Retrieval Module finds new candidates, we add them into the unlabelled candidate set. Lastly, because our algorithm does not want to search for candidates with relation keywords used in the sentence retrieval step again in future iterations, we mark them as "used".

### 3.2.2 Classification

In each iteration, our algorithm wants to present $b$ candidate facts $\{\langle S_j, R_i, O_j, T_j\rangle\}_{j=1...b}$ to human reviewers for assessments, where $b$ is the enterprise-defined batch size. In order to maximize the number of positive facts in the assessment results, rather than passing $b$ random candidates to human assessors, our Classification Module looks for $b$ most promising candidates from the unlabelled candidate set. A classifier is used to rank all candidates in the unlabelled candidate set from the most assuring one to the least assuring one. Specifically, for each candidate $\langle S_j, O_j, T_j\rangle$, the classifier predicts its likelihood of

[1] https://www.washingtonpost.com/blogs/post-partisan/wp/2017/10/25/amazon-key-is-silicon-valley-at-its-most-out-of-touch/

being an instance of the target relation $R_i$ based on all information we currently have (i.e., $Pr(\langle S_j, R_i, O_j, T_j \rangle = Positive)$), including initial positive instances from the algorithm input and labelled facts from previous iterations. If we know that the fact $\langle$*Microsoft, foundedBy, Bill Gates, "Bill Gates is a co-founder of Microsoft."*$\rangle$ has been labelled positive, then given an unlabelled candidate $\langle$*Microsoft, Paul G. Allen, "Paul G. Allen is a co-founder of Microsoft."*$\rangle$, our classifier will likely assign a high probability to it because its sentence structure is very similar to one of the existing positive fact. Once the prediction step has been completed for all candidates, our classifier will rank them based on their probabilities from the highest to the lowest. Now, the top $b$ candidate entity pairs with contexts $\{\langle S_j, O_j, T_j \rangle\}_{j=1...b}$ are the ones that our classifier is most confident about, so we convert them to candidate facts $\{\langle S_j, R_i, O_j, T_j \rangle\}_{j=1...b}$ by adding the target relation $R_i$ and pass them to human assessors. Details about how assessors annotate candidate facts are discussed in Section 3.1.3.

### 3.2.3 Feedback

After the batch of $b$ candidate facts is reviewed and labelled by human assessors, the assessment results will be taken back to the algorithm's Feedback Module. This module would produce positive facts to output, explore new relation keywords, and improve our classifier.

In the TRR algorithm, its output grows with each iteration rather than produced all at once after the algorithm is terminated. Every time when a batch of candidate facts are labelled by reviewers, the Feedback Module collects positive ones from the assessment results and append these facts to our algorithm output.

Because all candidates we work on are discovered using relation keywords in the Retrieval Module, it is essential to explore more keywords other than input keywords specified by the enterprise. A good relation keyword should be closely related to the target relation, which means that a sentence likely contains information about the target relation if it has the keyword. Our Feedback Module explores new relation keywords by looking for words frequently appearing in positive facts and have not been marked as "used" in previous iterations. As an illustration, if the target relation is $foundedBy$ and used relation keywords are {*founded, founder*}, given two positive facts $\langle$*Bose Corporation, foundedBy, Amar G. Bose, "Amar G. Bose is the founder of audio tech company Bose Corporation."*$\rangle$ and $\langle$*Salesforce.com, foundedBy, Marc Benioff, "The cloud computing company, Salesforce.com, was founded by Marc Benioff."*$\rangle$, the Feedback Module would observe that the word *company* appears in both sentences and has not been used before. In this case, *company* would be marked as our new relation keyword, and the Retrieval Module would use

it to look for new candidate entity pairs with contexts in the next iteration. There may be situations that no new relation keywords are explored from positive facts in the current iteration. In this case, our algorithm will skip the Retrieval Module in the next iteration since no word can be used to discover candidates.

Moreover, to enhance our classifier's accuracy, we could provide the classifier with more positive and negative examples by adding labelled facts to our training set. As the classifier gets more familiar with how positive/negative facts look like, its predictions on unlabelled candidates will become more accurate. For example, if a positive fact ⟨*Microsoft, foundedBy, Bill Gates, "Bill Gates is a co-founder of Microsoft."*⟩ is added to our training set, our classifier may learn that if an *Organization* entity and a *Person* entity appear together in a sentence "*Person* is a co-founder of *Organization*", they are likely expressing the relation *foundedBy*.

## 3.3 Implementation

In this thesis, we propose a Python system implementing the CAL-based algorithm discussed in Section 3.2. The overview of system implementation is listed in Algorithm 2. This section will introduce how each of the Retrieval, Classification, and Feedback Modules is implemented in our system.

### 3.3.1 Retrieval

As we discussed in Section 3.2.1, the Retrieval Module discovers new candidate entity pairs with contexts in two steps: sentence retrieval and candidate generation, with detailed steps listed in Algorithm 3.

Our system uses the IR toolkit Pyserini for sentence retrieval [9]. Given the algorithm input, we store each sentence in the given documents as a JSON object and the sentence's linked entity information as the object's fields. Then we index all JSON objects into Pyserini for future retrieval. Consider a sentence "The Washington Post owned by Amazon founder and chief executive Jeffrey P. Bezos."[2] with three entities {*(The Washington Post, Organization), (Amazon, Organization), (Jeffrey P. Bezos, Person)*}, we could store it as the following JSON object:

[2] https://www.washingtonpost.com/blogs/post-partisan/wp/2017/10/25/amazon-key-is-silicon-valley-at-its-most-out-of-touch/

**Algorithm 3:** The Retrieval Module Implementation.

Preparation. Store input documents and their linked entities' information as JSON objects, then index all JSON objects into Pyserini.

Step 1. For each unused relation keyword, call Pyserini's *search* function, with its parameter *query* being the keyword and parameter *number of hits to return* being infinity, to retrieve all sentences relevant to this keyword;

Step 2. Filter out sentences that do not satisfy the entity type requirements;

Step 3. Generate all valid candidate entity pairs with contexts from filtered sentences;

Step 4. Filter out candidate entity pairs with contexts who are known to be positive or negative from the prior information;

Step 5. Add filtered candidate entity pairs with contexts $\{\langle S_j, O_j, T_j \rangle\}_{j=1...m}$ to the unlabelled candidate set;

Step 6. Mark relation keywords used in step 1 as "used".

```
{
  "id": "doc1",
  "contents": "The Washington Post owned by Amazon founder and chief executive
        Jeffrey P. Bezos.",
  "entities": {
        "Organization": ["The Washington Post", "Amazon"],
        "Person": ["Jeffrey P. Bezos"]
            }
}.
```

In the sentence retrieval step, our system is able to retrieve all sentences relevant to each unused relation keyword by simply calling the Pyserini's *search* function with its parameter *query* being the relation keyword and *number of hits to return* being infinity, which returns a list of JSON objects ranked by their relevance scores. Since the target relation is defined with specific entity types, we could filter out objects that do not satisfy the type requirements. For example, if the specified entity types are *Organization* and *Person* entities, only JSON objects with at least one *Organization* and *Person* entity would be used to generate candidates in the next step.

Next, our system moves to the candidate generation step with all filtered JSON objects. There are no special techniques used in generating candidates, but multiple candidates can

be built from one sentence as discussed in Section 3.2.1, so our system ensures that all possible combinations are considered. Additionally, to avoid duplicate work, we want to filter out candidates $\langle S_j, O_j, T_j \rangle$ whose entities $S_j, O_j$ are known to be labelled positive or negative. There are two cases that a candidate that can be recognized as positive based on prior information. The first case is when its entity pair $S_j, O_j$ appears in a positive fact, we will know that $\langle S_j, O_j \rangle$ must satisfy the target relation. The other case is when the target relation is symmetric and its reversed entity pair $\langle O_j, S_j \rangle$ is included in a positive fact, then we will also know that the candidate $\langle S_j, O_j, T_j \rangle$ will be labelled positive. To illustrate, given a positive fact $\langle$*Hillary Clinton, spouse, Bill Clinton, "Hillary Clinton and Bill Clinton were married in 1975."*$\rangle$, we can learn that the entity pair $\langle$*Hillary Clinton, Bill Clinton*$\rangle$ satisfies the relation *spouse*, so there is no need to review any candidates $\langle$*Hillary Clinton, Bill Clinton,* $T_j\rangle$ or $\langle$*Bill Clinton, Hillary Clinton,* $T_j\rangle$ in future iterations. Moreover, a candidate $\langle S_j, O_j, T_j \rangle$ must be negative if the target relation $R_i$ is single-valued and an entity pair $\langle S_j, O_k \rangle$ appears in a positive fact, where $O_j \neq O_k$. For example, if the fact $\langle$*Bill Clinton, yearOfBirth, 1946, "Bill Clinton was born in 1946."*$\rangle$ has been labelled positive, we would know that any candidates with *Bill Clinton* and a *year* entity different from *1946* would always be labelled negative. It is worth mentioning that if a fact $\langle S_j, O_j, T_j \rangle$ has been labelled negative, it means that the sentence $T_j$ does not show the target relation exists between entities $S_j, O_j$, which does not necessarily imply that entities $S_j, O_j$ do not express the relation $R_i$. Thinking about a negative fact: $\langle$*Microsoft, foundedBy, Bill Gates, "Bill Gates has left Microsoft's board."*$\rangle$, we cannot say that the entity pair $\langle$*Microsoft, Bill Gates*$\rangle$ does not satisfy the *foundedBy* relation. After removing candidates whose entities are known to be positive or negative, our system adds rest candidates into the unlabelled candidate set, which will be ranked in the Classification Module. At the end, all relation keywords used in the sentence retrieval step are marked as "used".

### 3.3.2 Classification

The Classification Module wants to find $b$ most promising candidates from the unlabelled candidate set, which will be evaluated by human annotators. Algorithm 4 talks about how this module is implemented in our system. Specifically, it uses a logistic regression classifier to predict the probability of each candidate $\langle S_j, O_j, T_j \rangle$ expressing the target relation $R_i$ and then ranks them based on their predicted probabilities. Our system utilizes initial positive instances and all labelled facts as the training set.

At the beginning of this module, we randomly select 100 entity pairs with contexts from the corpus and temporarily add them to our negative training set. Then our module

computes feature sets for each fact $\langle S_j, R_i, O_j, T_j \rangle$ in the training set and each candidate $\langle S_j, O_j, T_j \rangle$ in the unlabelled candidate set. Our logistic regression classifier is trained using facts' feature sets. After the training step, we can remove the temporary 100 entity pairs with contexts from our training set. The next step is to use the trained classifier to estimate each unlabelled candidate $\langle S_j, O_j, T_j \rangle$'s probability of expressing the target relation $R_i$ and then ranks all candidates based on their probabilities from the highest to the lowest. Assuming human annotators examine $b$ candidate facts in each iteration, we extract the top $b$ candidates $\{\langle S_j, O_j, T_j \rangle\}_{j=1...b}$ from the ranked list, transform them to a batch of $b$ candidate facts $\{\langle S_j, R_i, O_j, T_j \rangle\}_{j=1...b}$ by adding the target relation information $R_i$, and pass them to human annotators for assessments.

**Algorithm 4:** The Classification Module Implementation.

Step 1. Extract 100 random entity pairs with contexts from the dataset and temporarily add them to the negative training set.
Step 2. Construct feature sets for facts in the training set and candidate entity pairs with contexts in the unlabelled candidate set;
Step 3. Train the logistic regression classifier using facts' feature sets;
Step 4. Remove the 100 random entity pairs with contexts added in Step 1;
Step 5. Estimate probabilities of candidate entity pairs with contexts being labelled positive by feeding their feature sets into the classifier;
Step 6. Rank all candidate entity pairs with contexts by their estimated probabilities;
Step 7. Transform the top $b$ candidate entity pairs with contexts $\{\langle S_j, O_j, T_j \rangle\}_{j=1...b}$ to $b$ candidate facts $\{\langle S_j, R_i, O_j, T_j \rangle\}_{j=1...b}$;
Step 8. Pass the batch of $b$ candidate facts $\{\langle S_j, R_i, O_j, T_j \rangle\}_{j=1...b}$ to annotators.

In our system, the feature set contains a flag indicating which entity comes first in $T_j$, TF-IDF scores, word vectors, lexical features, and syntactic features. We will talk about each feature in the rest of this section.

### Entity Positioning Flag

In order to recognize general patterns of a relation, given a candidate or fact, it is helpful to distinguish which entity, $S_j$ or $O_j$, comes first in its sentence $T_j$. Our system uses a flag indicating entities' positions, which will be true if entity $S_j$ comes first and be false otherwise. Consider a positive fact ⟨*Facebook, foundedBy, Mark Zuckerberg, "Facebook was founded by Mark Zuckerberg and fellow Harvard University students."*⟩, its flag would be assigned with a true value since *Facebook* comes before *Mark Zuckerberg* in the sentence.

### TF-IDF

The word-based TF-IDF (i.e., term frequency-inverse document frequency) score is utilized to measures how each word is relevant to the sentence $T_j$ among all sentences in the given corpus. Our system uses Pyserini's vectorizer to compute TF-IDF vectors. The vectorizer takes a list of indexed sentences as input and returns their corresponding TF-IDF vectors represented as sparse matrices. Pyserini also supports skipping stop words so that most common words (e.g., "the", "is", "in") can be ignored when computing TF-IDF values. For each candidate or fact, we call Pyserini's vectorizer with the skipping stop words option to convert its sentence $T_j$ to a list of TF-IDF scores, where each score represents the relevancy between a word and the sentence $T_j$. We expect sentences made of similar words to have similar TF-IDF values.

### Word Vectors

It is a common NLP technique representing a word as a multi-dimensional real-valued vector, where words with similar meanings are represented with similar vectors. Our system utilizes the open-source NLP library spaCy to map each sentence to a sentence vector [8]. The spaCy NLP pipeline takes a sentence as input and transforms it to a processed *Doc* object containing all linguistic features extracted from the given sentence, including the computed sentence vector. Specifically, spaCy first expresses each word in the sentence as a 300-dimensional word vector and then computes the 300-dimensional sentence vector by averaging all word vectors. With sentences being converted to vectors of real numbers, the similarity between sentences can be determined using the difference between their sentence vectors.

### Lexical Features

We also use spaCy to collect lexical features from sentences. Besides word vectors, spaCy's processed *Doc* objects also contain various lexical features, such as lemmas, POS tags, and word shapes. Among all available properties, our system selects lexical features similar to those used in previous relation extraction work [12]:

- Lemmas and POS tags of the left three words to the entity appearing first
- Lemmas and POS tags of the right three words to the entity appearing second
- Lemmas and POS tags of all middle words between two entities

All features we talked about above are unigram, which means each word is analyzed independently. For instance, when our system looks at a two-word phrase *high school*, it would see *high* and *school* as two separate words rather than recognizing them together as one phrase. To catch more information from potential two-word phrases, we also consider bigram lexical features as follows:

- Lemmas and POS tags of consecutive two words between two entities

### Syntactic Features

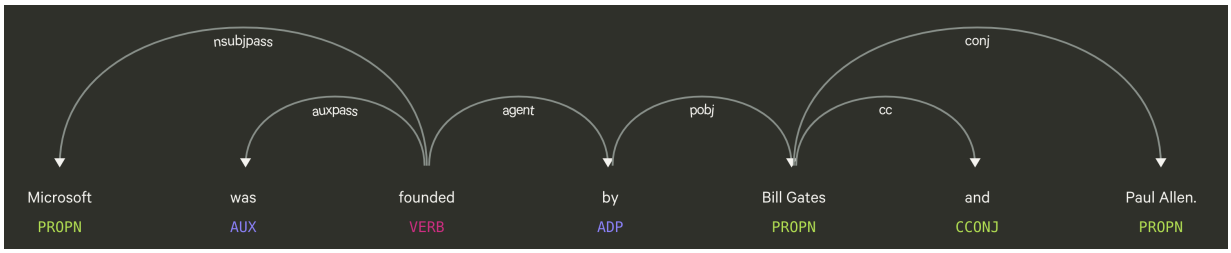


Figure 3.4: Dependency graph generated by spaCy.

The last part of the feature set is made by syntactic features, which are also obtained from spaCy's *Doc* objects. Given the sentence "Microsoft was founded by Bill Gates and Paul Allen.", Figure 3.4[3] explicates its dependency graph generated by spaCy. Each arc connects a head word and a child word, with the arc label being dependency type. For

[3] https://explosion.ai/demos/displacy

example, the left-most arc tells us that *founded* is the head word of *Microsoft* with the dependency type *nsubjpass*. Each word may have multiple child nodes but at most one head node. In our system, syntactic features include:

- Lemma, POS tag and dependency type of the head word of entity $S$
- Lemmas, POS tags and dependency types of all child words of entity $S$
- Lemma, POS tag and dependency type of the head word of entity $O$
- Lemmas, POS tags and dependency types of all child words of entity $O$

As an illustration, assuming we have a positive fact $\langle$*Microsoft, foundedBy, Bill Gates, "Microsoft was founded by Bill Gates and Paul Allen."*$\rangle$, then its feature set is shown in Table 3.1, where $s_1, ..., s_6$ are computed TF-IDF scores and $v_1, ..., v_{300}$ are values in the 300-dimensional sentence vector. To feed the feature set into our logistic regression classifier, the entity positioning flag, lexical features, and syntactic features are converted to indicator variables. If a specific feature is not included in the feature set, it means that the feature does not apply to this specific sentence. For example, entity *Microsoft* is the first word in the given sentence, so it does not have the left three words' information which lexical features need.

### 3.3.3 Feedback

As discussed in Section 3.2.3, the Feedback Module mainly works on three things: contributing to output, exploring new relation keywords, and enriching the training set. Algorithm 5 describes the implementation of this module.

**Algorithm 5:** The Feedback Module Implementation.

Step 1. Find all positive facts $\{\langle S_j, R_i, O_j, T_j\rangle\}_{j=1...p}$ in the assessment results and output them;
Step 2. Mark a word as a relation keyword if its frequency of appearing in positive facts exceeds a pre-defined threshold and has not been marked as "unused";
Step 3. Mark the most common unused word in positive facts as relation keyword if the unlabelled candidate set exhausts (terminate the system if all words in positive facts have already been marked as "used" in previous iterations);
Step 4. Add labelled facts to the training set.

After human assessments are finished, the assessment results are taken back to our system. The Feedback Module will collect all facts that were labelled positive and appends them to system output. In other words, our system output is incrementally built over each iteration. The enterprise does not need to terminate the system to see relation facts; instead, our system presents facts to the enterprise immediately after they are coded as "positive" by reviewers.

This module also looks for new keywords to retrieve more candidates in future batches. It is worth mentioning that our system ignores words that have been marked as "used" and all stop words in keyword exploration. In our implementation, there are two cases that a new keyword would be discovered. The first case is when a word appears in positive facts' sentences with a frequency higher than our pre-determined frequency threshold, and we would mark the word as our new keyword. For instance, with a frequency threshold 0.5 and $n$ positive facts $\{\langle S_j, R_i, O_j, T_j\rangle\}_{j=1...n}$, if a word is contained in at least $0.5n$ different $T_j$'s and it is not one of our "used" relation keywords, the system would mark it as a new keyword. The second case is when we have labelled all promising candidates in the unlabelled candidate set, it is urgent to explore more candidates. In this case, the most common unused word in positive facts' sentences would be chosen as the new keyword regardless of its frequency. There is a special scenario that all promising candidates in the unlabelled candidate set are evaluated, and all words in positive facts' sentences have also been used as relation keywords. Then we would terminate our system promptly since no word can be used to explore new candidates anymore. Overall, if neither of the two keyword exploration conditions were met, no keyword would be learned in this iteration, meaning that no word can be used to discover new candidates in the next iteration. Hence, our system would skip the next Retrieval Module and directly proceed to the Classification Module.

Another critical component of the Feedback Module is to append all labelled facts to the corresponding positive and negative training set, which will further train the logistic regression classifier in the next iteration's Classification Module.

| Category | Feature |
|---|---|
| Flag | "entityS_first": 1 |
| TF-IDF | "Microsoft": $s_1$<br>"founded": $s_2$<br>"Bill": $s_3$<br>"Gates": $s_4$<br>"Paul": $s_5$<br>"Allen": $s_6$ |
| Word Vectors | "wv1": $v_1$<br>"wv2": $v_2$<br>...<br>"wv299": $v_{299}$<br>"wv300": $v_{300}$ |
| Unigram Lexical | "ent_right1_lemma_and": 1<br>"ent_right1_pos_CCONJ": 1<br>"ent_right2_lemma_Paul": 1<br>"ent_right2_pos_PROPN": 1<br>"ent_right3_lemma_Allen": 1<br>"ent_right3_pos_PROPN": 1<br>"ent_mid_lemma_be": 1<br>"ent_mid_pos_AUX": 1<br>"ent_mid_lemma_found": 1<br>"ent_mid_pos_VERB": 1<br>"ent_mid_lemma_by": 1<br>"ent_mid_pos_ADP": 1 |
| Bigram Lexical | "ent_bi_mid_lemma_Microsoft_be": 1<br>"ent_bi_mid_pos_PROPN_AUX": 1<br>"ent_bi_mid_lemma_be_found": 1<br>"ent_bi_mid_pos_AUX_VERB": 1<br>"ent_bi_mid_lemma_found_by": 1<br>"ent_bi_mid_pos_VERB_ADP": 1<br>"ent_bi_mid_lemma_by_Bill": 1<br>"ent_bi_mid_pos_ADP_PROPN": 1 |
| Syntactic | "entS_head_lemma_found": 1<br>"entS_head_pos_VERB": 1<br>"entS_head_dep_nsubjpass": 1<br>"entO_head_lemma_by": 1<br>"entO_head_pos_ADP": 1<br>"entO_head_dep_pobj": 1<br>"entO_child_lemma_and": 1<br>"entO_child_pos_CCONJ": 1<br>"entO_child_dep_cc": 1<br>"entO_child_lemma_Paul_Allen": 1<br>"entO_child_pos_PROPN": 1<br>"entO_child_dep_conj": 1 |

Table 3.1: An example of features extracted.

# Chapter 4

# Evaluation

## 4.1 Experimental Setup

In this chapter, we evaluate the effectiveness of our proposed system. In the TRR problem definition discussed in Section 3.1, TRR takes enterprise data and a specified relation as input, assuming that all entities in the enterprise data are recognized and linked to their internal KBs. However, we do not have the internal enterprise KB, their linked data, or any specified relation, which means all of them have to be simulated as the experiment input. In specifically, we use Wikidata as the enterprise KB and use two corpora of news articles, TREC Washington Post Corpus and The New York Times Annotated Corpus, as the enterprise data. For the input relation, we choose 12 relations of four entity types defined on Wikidata and commonly used in real life. Furthermore, in order to reduce the cost of manual work in human assessments, we automate the assessment process by labelling relation candidates using the knowledge of Wikidata. This section talks about how we simulate the system input and evaluate it against Wikidata in detail. At the end of the section, we also specify some pre-defined parameters for our experiments, such as features used to train the classifier and the frequency threshold used in exploring keywords.

### 4.1.1 Input

#### Dataset

We simulate the enterprise KB with Wikidata,[1] which contains 92,678,133 entities of many different types, including *Person*, *Organization*, *Event*, etc. Besides, the enterprise data is simulated with TREC Washington Post Corpus[2] (WaPost) and The New York Times Annotated Corpus[3] (NYTimes). The WaPost is constructed by 595,037 news articles and blog posts between January 2012 and August 2017, and the NYTimes consists of 1,855,650 New York Times articles from January 1, 1987, to June 19, 2007.

In the problem assumption, the enterprise data has all its entities recognized and linked to the KB. Hence, we would like to first find all entities in the WaPost and NYTimes corpora and then link these entities to Wikidata items. Specifically, spaCy NER is used to identify entities. Since our system extracts relation instances at sentence-level, we use spaCy's sentencizer to split all WaPost and NYTimes articles into sentences, which gives us 29,833,284 sentences from the WaPost corpus and 64,693,394 sentences from the NYTimes corpus. Then we apply spaCy's trained NLP pipeline to extract NLP features from each sentence [8]. As shown in Figure 2.4, the spaCy pipeline involves an NER, which can recognize multiple categories of entities from texts, such as *Person* and *Organization* entities. If there is any entity type that we are interested in but is not covered by the spaCy NER, we can always add customized entity classes to the NER model using spaCy's training library. After the NER model helps us identify all entities from the WaPost and NYTimes sentences, we are ready to connect them to Wikidata items using the entity linking REL [21]. REL takes a sentence and its NER information as input, and then it looks for the corresponding Wikidata item for each entity in the NER results. For example, given a WaPost sentence "The Washington Post is owned by Amazon founder and chief executive Jeffrey P. Bezos",[4] three entities are recognized and linked to Wikidata using spaCy NER and REL: *(The Washington Post, Organization, Q166032)*, *(Amazon, Organization, Q3884)*, and *(Jeffrey P. Bezos, Person, Q312556)*, where each entity is represented in the format *(Mention Text, NER Type, Wikidata QID)*. However, given the unavoidable NER error and entity linking error, our simulated enterprise KB cannot satisfy the problem assumption perfectly. For example, given a sentence "Routhouska served

[1] https://www.wikidata.org/wiki/Wikidata:Main_Page

[2] https://trec.nist.gov/data/wapost/

[3] https://catalog.ldc.upenn.edu/LDC2008T19

[4] https://www.washingtonpost.com/blogs/post-partisan/wp/2017/10/25/amazon-key-is-silicon-valley-at-its-most-out-of-touch/

as assistant principal of Ramsay in the 2011-2012 school year.",[5] spaCy NER recognizes *Ramsay* as a *Person* entity mistakenly while it is actually an *Organization* entity. Consider another sentence "Brzezinski was born in 1932 in Switzerland.",[6] REL links *Brzezinski* to Wikidata item *Zbigniew Brzezinski (Q168041)*, but it actually refers to *Emilie Benes Brzezinski (Q2233935)*.

In our experiments, the system is evaluated with relations of four different entity pair types: ⟨*Organization, Year*⟩, ⟨*Person, Year*⟩, ⟨*Organization, Person*⟩ and ⟨*Person, Person*⟩. We count each pair's occurrence in the WaPost and NYTimes sentences, and the results are displayed in Table 4.1. In both WaPost and NYTimes corpora, the occurrences of different types follow the same trend that ⟨*Organization, Year*⟩, ⟨*Person, Year*⟩, ⟨*Organization, Person*⟩ and ⟨*Person, Person*⟩ appear in an ascending number of times, which means that news articles tend to talk more about the connection between two people or between people and organizations than the connection between years and people/organizations. In addition, all entity pairs appear more times in the NYTimes corpus than the WaPost corpus because the NYTimes dataset contains around three times of articles as the WaPost dataset does.

| Corpus | ⟨Organization, Year⟩ | ⟨Person, Year⟩ | ⟨Organization, Person⟩ | ⟨Person, Person⟩ |
|---|---|---|---|---|
| WaPost | 655,893 | 762,178 | 3,343,061 | 6,026,994 |
| NYTimes | 1,483,988 | 2,003,612 | 6,665,800 | 19,681,115 |

Table 4.1: Different types of entity pairs from the WaPost and NYTimes corpora.

### Relations

As demonstrated in Table 4.2, 12 relations across four entity pair types are studied in our experiments. All relations are selected from Wikidata, and their IDs come from Wikidata property PID. For instance, the *foundedBy* relation is of type ⟨*Organization, Person*⟩ with Wikidata PID *P112*.[7] As mentioned in Section 3.1.2, the enterprise also needs to provide some related information for their specified relations. Table 4.3 indicates relation properties (single-valued vs. multi-valued and symmetric vs. asymmetric) and 2–5 relation keywords

[5] https://www.washingtonpost.com/local/alexandria-arlington-news-in-brief/2015/07/07/4e29191a-1f36-11e5-aeb9-a411a84c9d55_story.html

[6] washingtonpost.com/lifestyle/magazine/emilie-brzezinski-is-her-own-kind-of-power-player-an-artist-with-a-chainsaw/2014/08/14/d8b3aa2a-af73-11e3-a49e-76adc9210f19_story.html

[7] https://www.wikidata.org/wiki/Property:P112

| Type | ID | Name | Examples |
|---|---|---|---|
| ⟨Person, Person⟩ | P22 | father | ⟨George W. Bush, George H. W. Bush⟩ |
| | P25 | mother | ⟨George W. Bush, Barbara Bush⟩ |
| | P26 | spouse | ⟨George H. W. Bush, Barbara Bush⟩ |
| | P3373 | sibling | ⟨Jeb Bush, George W. Bush⟩ |
| | P40 | child | ⟨George H. W. Bush, George W. Bush⟩ |
| ⟨Organization, Person⟩ | P112 | foundedBy | ⟨Apple Inc., Steve Jobs⟩ |
| | P169 | CEO | ⟨Apple Inc., Tim Cook⟩ |
| | P69 | educatedAt | ⟨Duke University, Tim Cook⟩ |
| ⟨Person, Year⟩ | P26P580 | yearOfMarriage | ⟨George H. W. Bush, 1945⟩ |
| | P569 | yearOfBirth | ⟨George H. W. Bush, 1924⟩ |
| | P570 | yearOfDeath | ⟨George H. W. Bush, 2018⟩ |
| ⟨Organization, Year⟩ | P571 | yearFounded | ⟨Apple Inc., 1976⟩ |

Table 4.2: Relations used in experiments.

| Name | SingleValued | Symmetry | Keywords |
|---|---|---|---|
| father | True | False | father, son, daughter |
| mother | True | False | mother, son, daughter |
| spouse | False | True | wife, husband, fiancee, husband |
| sibling | False | True | sister, brother |
| child | False | False | son, daughter, mother, father, children |
| foundedBy | False | False | founder, owner, founded |
| CEO | False | False | ceo, chief executive |
| educatedAt | False | False | university, college, undergraduate, graduate, Ph.D |
| yearOfMarriage | False | False | wedding, marriage |
| yearOfBirth | True | False | born, lived |
| yearOfDeath | True | False | died, death, killed, suicide, lived |
| yearFounded | True | False | founded, established, built, began, opened |

Table 4.3: Relations and their related information.

for each relation. In addition, we look for 10 positive instances from the given dataset for each relation, where all of them are listed in Appendix A.

### 4.1.2 Automated Human Assessment

In our experiment, the human assessment process is automated with Wikidata. Given a target relation $R_i$ and a candidate fact $\langle S_j, R_i, O_j, T_j \rangle$, if the relation $R_i$ exists between two entities $S_j, O_j$ on Wikidata, this fact will be labelled positive automatically; otherwise, the fact will be labelled negative. For example, given a candidate fact ⟨*Facebook (Q380), foundedBy (P112), Mark Zuckerberg (Q36215), "Facebook founder Mark Zuckerberg more than doubled his net worth, to $28.5 billion, and ranked 21st."*[8]⟩, we want to verify if entities *Facebook* and *Mark Zuckerberg* satisfy the relation *foundedBy* on Wikidata. Since entity QIDs and the relation PID are known, we retrieve values associated to property *P112* from the entity *Q380*[9] on Wikidata, which give us Facebook's five founders *Mark Zuckerberg (Q36125)*, *Eduardo Saverin (Q312663)*, *Dustin Moskovitz (Q370217)*, *Chris Hughes (Q370321)* and *Andrew McCollum (Q4757939)*. As our entity *Mark Zuckerberg (Q36125)* is one of the retrieved values, we label this fact as positive.

However, the incompleteness of Wikidata may lead to false negatives. Given a candidate fact ⟨*Fundable (Q5508894), foundedBy (P112), Wil Schroter (Q8000023), "Wil Schroter, co-founder and chief executive of Fundable.com, an equity crowdfunding platform for startups based in Columbus, Ohio."*[10]⟩, it is obvious for a human reviewer that entities *Fundable* and *Wil Schroter* satisfy the target relation *foundedBy* by reading the given sentence. However, there is no *P112* property information on Wikidata item *Q5508894*, so our automated evaluation process would mistakenly label this fact as negative. To avoid this type of error, our automated annotator is revised to label the candidate fact as "unknown" rather than "negative" if the target relation property is not available for the fact's entity $S_j$ on Wikidata. These facts are neither considered positive nor negative, so we prevent adding false negative facts to our training set in the Feedback Module. Algorithm 6 concludes how the automated process labels candidate facts.

Assessment results generated by the automated process can be different from results manually labelled by human reviewers. Consider a candidate fact ⟨*Ekaterina Gordeeva*

[8] https://www.washingtonpost.com/business/economy/national-business-and-economy-roundup/2014/03/03/f984abce-a31a-11e3-8466-d34c451760b9_story.html

[9] https://www.wikidata.org/wiki/Q380

[10] https://www.washingtonpost.com/business/on-small-business/heres-what-small-business-owners-want-from-washington-in-2014/2013/12/31/269af556-69a9-11e3-ae56-22de072140a2_story.html

**Algorithm 6:** The Automated Human Assessment Implementation.

Input. A batch of $b$ candidate facts $\{\langle S_j, R_i, O_j, T_j\rangle\}_{j=1...b}$ where entities $S_j, O_j$ are linked to Wikidata items $Q_{S_j}, Q_{O_j}$ and relation $R_i$ is linked to Wikidata property $P_{R_i}$.

Step 1. Set $k$ to 1;

Step 2. If the Wikidata item $Q_{S_k}$ has property $P_{R_i}$, retrieve values associated to the property and move to Step 3; otherwise, label the fact $\langle S_k, R_i, O_k, T_k\rangle$ as "unknown" and move to Step 4;

Step 3. If $Q_{O_k}$ matches one of the property values, label the fact $\langle S_k, R_i, O_k, T_k\rangle$ as "positive"; otherwise, label the fact as "negative";

Step 4. Increment $k$ by 1;

Step 5. Repeat steps 2 through 4 until all $b$ candidate facts are labelled;

*(Q254053), spouse (P26), Sergei Grinkov (Q465258), "She had helped to guide Ekaterina Gordeeva and Sergei Grinkov, two-time Olympic champions largely regarded as the best ever – and she had a strategy for new American dominance."*[11]$\rangle$, our automated process will mark this fact as positive because entities *Ekaterina Gordeeva* and *Sergei Grinkov* satisfy the *spouse* relation on Wikidata. However, human reviewers are asked to code a fact $\langle S_j, R_i, O_j, T_j\rangle$ as positive if and only if the sentence $T_j$ proves that the target relation $R_i$ exists between two entities $S_j, O_j$. Since the sentence in this example does not provide information about the *spouse* relation between *Ekaterina Gordeeva* and *Sergei Grinko*, human reviewers should label this fact as negative. Thus, the automated assessment process would produce false positives in the case that the relation between entities can be verified on Wikidata but is not expressed by the fact's sentence.

### 4.1.3 Other Parameters

In this section, we talk about some parameters used in our experiments. The first parameter is the feature set used to train the logistic regression classifier in the Classification Module. In our primary experiments, the feature set for candidates and facts is made of the entity positioning flag, TF-IDF scores, a union of unigram and bigram lexical features, and syntactic features. We will run ablation experiments later to evaluate the effectiveness of other feature groups. As for the next parameter, our experiment sets the batch size $b$ as 6,000, which means in each iteration, a batch of 6,000 candidate facts are reviewed and

[11] https://www.washingtonpost.com/news/olympics/wp/2016/03/30/for-americans-at-world-figure-skating-championships-it-may-be-all-about-the-dance/

labelled in human assessments. Hence, the 6,000 most promising candidate entity pairs with contexts among all unlabelled ones are extracted in the Classification Module and passed to the human assessment process. The third parameter is the frequency threshold used to explore relation keywords in the Feedback Module. We set the threshold as 0.1 so that a word would be marked as a new relation keyword if it appears in at least 10% of positive facts' sentences. Lastly, the termination of our system is decided by the enterprise. Theoretically, the enterprise would terminate the system whenever they believe that enough positive facts have been produced. However, there is no such decision maker in our experiments, so we have to simulate one. We use the number of facts labelled as the termination criterion. Specifically, the system would terminate if 300,000 facts have been labelled. Since per candidate fact $\langle S_j, R_i, O_j, T_j \rangle$ contains exactly one sentence $T_j$ from the dataset, that is to say, once 1.01% of WaPost sentences or 0.46% of NYTimes sentences have been reviewed, our system would stop.

## 4.2 Evaluation Metrics

For each relation $R$, the recall can be calculated after each iteration using the following equation

$$\text{recall} = \frac{\text{\# of positive facts identified so far}}{\text{\# of ground truth for } R \text{ in the corpus}}, \tag{4.1}$$

which represents the proportion of candidate facts labelled as positive so far to all instances of the target relation that we should be able to recognize by reading the entire corpus (i.e., ground truth). To determine the number of ground truth for a relation $R$, we could count how many unique entity pairs in the WaPost and NYTimes sentences satisfy the target relation $R$ on Wikidata. For example, given a sentence from the WaPost corpus "The Washington Post is owned by Jeffrey P. Bezos, the founder and chief executive of Amazon.",[12] entities *Amazon*[13] and *Jeffrey P. Bezos*[14] satisfy the *foundedBy* relation on Wikidata, so $\langle$*Amazon, Jeffrey P. Bezos*$\rangle$ is counted as a ground truth for *foundedBy* in the WaPost corpus. Table 4.4 shows the number of ground truth for each relation in the WaPost corpus and the NYTimes corpus. We can observe that all relations' instances rarely appear in both datasets with all prevalence rates less than $2 \times 10^{-4}$. It is worth noting that due to the NER/entity linking error and the false positives/negatives problem

[12] https://www.washingtonpost.com/blogs/post-partisan/wp/2017/10/25/amazon-key-is-silicon-valley-at-its-most-out-of-touch/

[13] https://www.wikidata.org/wiki/Q3884

[14] https://www.wikidata.org/wiki/Q312556

mentioned in Section 4.1.1 and 4.1.2, the number of ground truth calculated here may differ from the actual number of relation instances that we could obtain by reading all sentences. To illustrate, the candidate fact ⟨*Ekaterina Gordeeva, spouse, Sergei Grinkov, "She had helped to guide Ekaterina Gordeeva and Sergei Grinkov, two-time Olympic champions largely regarded as the best ever – and she had a strategy for new American dominance."*[15]⟩ mentioned in Section 4.1.2 would be counted as a ground truth by mistake.

| Relation | WaPost | | NYTimes | |
|---|---|---|---|---|
| | # Ground Truth | Prevalence ($10^{-4}$) | # Ground Truth | Prevalence ($10^{-4}$) |
| P22 father | 831 | 0.2785 | 2226 | 0.3441 |
| P25 mother | 311 | 0.1042 | 770 | 0.1190 |
| P26 spouse | 1400 | 0.4693 | 3245 | 0.5016 |
| P3373 sibling | 674 | 0.2259 | 1502 | 0.2322 |
| P40 child | 1155 | 0.3872 | 3032 | 0.4687 |
| P112 foundedBy | 1420 | 0.4760 | 2926 | 0.4523 |
| P169 CEO | 383 | 0.1284 | 337 | 0.0521 |
| P69 educatedAt | 3493 | 1.1708 | 12231 | 1.8906 |
| P26P580 yearOfMarriage | 1141 | 0.3825 | 2557 | 0.3952 |
| P569 yearOfBirth | 2184 | 0.7321 | 9865 | 1.5249 |
| P570 yearOfDeath | 3139 | 1.0522 | 9944 | 1.5371 |
| P571 yearFounded | 2150 | 0.7207 | 6293 | 0.9727 |

Table 4.4: Ground truth statistics.

We consider the final recall, the recall graph, and the recall @ $aN$ as evaluation metrics. At the end of the experiment, we calculate the final recall using Equation 4.1, where the numerator is the total number of positive facts in the system output, and the denominator is the number of ground truth. The recall graph and recall @ $aN$ are similar to the gain curve and recall @ $aR + b$ metric in the TREC Total Recall Track discussed in Section 2.1.4. We plot the recall curve for each experiment, where the $x$-axis denotes the number of facts labelled, with a percentage showing the proportion of reviewed sentences to all sentences in the corpus, and the $y$-axis denotes the temporary recall after labelling $x$ facts. Hence, the final recall can be read from the $y$ value of the recall curve's ending point, and the curve also tells us how much the recall increases per iteration. The recall @ $aN$ is defined as the recall after $aN$ candidate facts are labelled, where $a$ can be any real number and $N$ is the number of ground truth for the target relation $R$. Our experiments consider parameter $a \in \{3, 6, 12, 24\}$. For example, when we look for instances for the relation *foundedBy* from the WaPost corpus, the recall @ $3N$ represents the recall after labelling

[15] https://www.washingtonpost.com/news/olympics/wp/2016/03/30/for-americans-at-world-figure-skating-championships-it-may-be-all-about-the-dance/

4,260 (i.e., $3 \times 1,420$) facts, which can also be seen as the recall after reviewing 4,260 sentences. Note that positions where $aN$ facts get labelled for all $a \in \{3, 6, 12, 24\}$ are marked as red vertical dashed lines in recall graphs.

## 4.3 Results and Analyses

### 4.3.1 Primary Results

We run experiments with 12 relations mentioned in Section 4.1.1 on WaPost and NYTimes corpora. In our experiment, manual assessments are replaced by the automated labelling process using the knowledge of Wikidata. As discussed in Section 4.1.3, the original feature set used to train our classifier consists of TF-IDF scores, unigram and bigram lexical features, and syntactic features. Moreover, every experiment sets the batch size to 6,000 and the keyword frequency threshold to 0.1. Once 300,000 candidate facts are labelled, the experiment would be terminated.

#### WaPost Results

| Relation | P22 | P25 | P26 | P3373 | P40 | P112 | P169 | P69 | P26P580 | P569 | P570 | P571 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Recall(%) | 81.83 | 83.60 | 77.71 | 75.82 | 83.29 | 79.37 | 95.56 | 87.43 | 99.74 | 98.76 | 99.27 | 99.07 |

Table 4.5: Final recalls of 12 relations from the WaPost corpus.

Table 4.5 shows final recalls for all relations over the WaPost corpus, which are between 75.82% to 99.74%, with an average recall 88.45% and median 85.52%. To put it another way, on average, our system can find 88.45% of the target relation's instances by reviewing only 1.01% of sentences in the corpus. We investigate the relation ground truth our system missed and find that a majority of them are false ground truth. In specific, we select one relation from every entity pair type: *mother (P25)*, *CEO (P169)*, *yearOfMarriage (P26P580)*, and *yearFounded (P571)*. Then we look at each of their missing ground truth. It turns out that 62.75% of *P25* missing instances, 100% of *P169* missing instances, 100% of *P26P580* missing instances, and 85% of *P571* missing instances should not be counted towards their ground truth. These instances are incorrectly added to ground truth because the relation information can be verified on Wikidata, but it is not expressed by any sentence in the corpus. For instance, a false ground truth for the relation *yearOfMarriage* is ⟨*George H.W. Bush, yearOfMarriage, 1945, "In that sense, George H.W. Bush's proclamation of a*

*'new world order' in 1991 was really a statement about the continuation of the post-1945 order."*[16]⟩. *George H.W. Bush*'s marriage year is *1945* on Wikidata, but we are not able to obtain this information by reading the sentence itself. Therefore, actual final recalls would be higher than recalls achieved in our experiments if we could eliminate all false ground truth.

Figure 4.1 displays relations' recall curves. Most graphs (except *P26P580*, *P569*, *P570* and *P571*) have the same shape that increases rapidly at the beginning, and the increasing speed decreases gradually until the recall curve plateaus between 75% and 100%. In each iteration, our classifier selects 6,000 candidate entity pairs with contexts, which have the highest likelihoods of being positive, from the unlabelled candidate set and passes them to the labelling process as candidate facts. When all high-likelihood candidate facts have been labelled, the simulated assessor will be presented with low-likelihood candidate facts, and this is when the increasing speed starts to slow down. Furthermore, a newly discovered keyword may result in a jump-up at the recall curve. After we have labelled the most assuring candidate facts, our recall curve would grow slowly or even stop growing. When a new keyword is identified in the Feedback Module, our system will call the Retrieval Module to look for unseen candidate entity pairs with contexts using the new keyword in the next iteration, where some of these candidates may be positive instances of the target relation. Therefore, our recall would be boosted when the lately retrieved candidates got labelled positive, leading to a jump-up at the recall curve. For example, when extracting instances of the relation *sibling (P3373)*, our model discovers a new keyword *son* after reviewing 152,539 sentences. Hence, a recall jump is observed in our *P3373* recall graph between 150,000 to 200,000 reviewed sentences.

Besides, recall graphs for relations *P26P580*, *P569*, *P570*, and *P571* are different from others in two perspectives. The first difference is that their curves end at the middle of graphs, while other curves proceed to the right boundary of their $x$-axes. As discussed in Section 3.3.3, when all candidate entity pairs with contexts in the unlabelled candidate set were examined, and all words in positive facts' sentences were "used", our system would terminate immediately. By looking at Table 4.1, there are much fewer potential entity pairs for these four relations than for other relations. Specifically, P26P580, P569, and P570 have 762,178 entity pairs of type ⟨*Person, Year*⟩, and P571 have only 655,893 entity pairs of type ⟨*Organization, Year*⟩. Therefore, it is not a surprise that all candidate entity pairs with contexts and words are exhausted before 300,000 facts get evaluated. At that point, the experiment would terminate right away, making the recall curve halt at the middle of the graph. The other distinction is that these four relations' recall curves

[16] https://www.washingtonpost.com/news/in-theory/wp/2016/01/29/politicians-say-american-leadership-is-in-decline-theyre-wrong/

continue rising at the end rather than reaching plateaus. To determine the cause of the shape difference, we take a close look at their later-staged iterations. It turns out that new keywords are explored every iteration because there are not enough promising candidate entity pairs with contexts left in our unlabelled candidate set, which results in jump-ups on the recall curves. Indeed, since the recall curve "jumps" in each iteration, it becomes the continuously increasing shape as what we see.

The recalls @ $aN$ for $a \in \{3, 6, 12, 24\}$ are shown in Table 4.6, where mean recalls at $3N, 6N, 12N, 24N$ are 22.82%, 33.66%, 46.08%, and 58.62% respectively. Average recall @ $3N$ being 22.82% means that our system is able to retrieve 22.82% of relation ground truth after labelling 3 times as many ground truth in total, which indicates the precision is around 7.61% (i.e., 22.82%/3). To figure out why the precision is quite low, we take a look at the first 100 labelled facts for the *foundedBy (P112)* relation. In the automated assessment results, 28, 30, 42 facts are labelled positive, negative, and unknown correspondingly, leading to a 28% precision among these 100 labelled facts. Then we manually annotate these candidate facts and get 58 positive ones and 42 negative ones, which doubles the precision from 28% to 58%. There are two major factors making the automatic process produce incorrect labels. The first factor is the incompleteness of Wikidata. Given a candidate fact ⟨*Sherpa Ventures (Q28405610), foundedBy (P112), Shervin Pishevar (Q20987648), "Shervin Pishevar, the co-founder and co-chief executive of San Francisco venture capital firm Sherpa Ventures, tweeted Tuesday that he would begin and fund a 'legitimate campaign' to help the world's sixth-largest economy become its own nation, 'New California.'"*[17]⟩, our automated process assigns the "unknown" label to it since the *foundedBy* property does not exist on the Wikidata item *Sherpa Ventures*. Nonetheless, it should be labelled positive as the given sentence expresses the target relation. The other factor is the entity linking error. For example, the candidate fact ⟨*Microsoft (Q2283), foundedBy (P112), Allen (Q5272620), "As a child, Allen, the co-founder of Microsoft, knew the names of the Mercury 7 astronauts, as if they were the star players of his favorite baseball team."*[18]⟩ is labelled as negative by the automated process because the entity *Allen* is linked to *Dick Allen (Q5272620)* instead of the Microsoft co-founder *Paul Garder Allen (Q162005)*. Due to the incompleteness of Wikidata information and the incorrect entity linking, some of the candidate facts labelled as negative or unknown by the automated process are actually positive facts. Hence, if our problem assumption about linked entities is fully satisfied and

[17] https://www.washingtonpost.com/politics/while-the-country-shifts-to-the-right-california-continues-to-move-left/2016/11/10/1c6cc602-a6d9-11e6-ba59-a7d93165c6d4_story.html

[18] https://www.washingtonpost.com/business/economy/the-billionaire-space-barons-and-the-next-giant-leap/2016/08/19/795a4012-6307-11e6-8b27-bb8ba39497a2_story.html

the labelling process is completed by human reviewers, the results' precision should be higher than what we currently have.

| Relation | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| P22 father | 6.50 | 13.00 | 27.91 | 56.81 |
| P25 mother | 3.65 | 7.30 | 14.60 | 27.40 |
| P26 spouse | 19.05 | 34.84 | 54.91 | 64.31 |
| P3373 sibling | 11.25 | 22.50 | 38.86 | 53.14 |
| P40 child | 10.65 | 21.42 | 42.01 | 61.85 |
| P112 foundedBy | 28.30 | 43.85 | 55.70 | 63.95 |
| P169 CEO | 11.80 | 23.60 | 47.20 | 68.01 |
| P69 educatedAt | 21.66 | 38.74 | 61.69 | 75.57 |
| P26P580 yearOfMarriage | 19.50 | 34.22 | 35.82 | 39.42 |
| P569 yearOfBirth | 61.56 | 64.55 | 65.83 | 67.49 |
| P570 yearOfDeath | 46.55 | 58.71 | 61.64 | 64.93 |
| P571 yearFounded | 33.38 | 41.18 | 46.82 | 60.55 |

Table 4.6: Recall @ aN for the WaPost corpus.

### NYTimes Results

| Relation | P22 | P25 | P26 | P3373 | P40 | P112 | P169 | P69 | P26P580 | P569 | P570 | P571 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Recall(%) | 78.35 | 74.94 | 67.58 | 63.72 | 77.18 | 73.10 | 90.80 | 77.30 | 59.64 | 64.78 | 71.41 | 89.61 |

Table 4.7: Final recalls of 12 relations from the NYTimes corpus.

Final recalls for the NYTimes corpus is displayed in Table 4.7, ranging from 59.64% to 90.80%. The average of final recalls is 74.03%, which is very close to their median 74.02%. Thus, by reading 0.46% of NYTimes sentences, our system could find 74.03% of a specific relation's instances on average. The NYTimes dataset contains more than three times of articles as the WaPost does, and there is also more ground truth for each relation in the NYTimes than in the WaPost corresponding to Table 4.4, leading to a larger denominator in the recall equation 4.1. However, our system terminates after labelling a fixed number of candidate facts, so it is expected that the NYTimes experiments' final recalls are lower than the WaPost experiments'.

Moreover, recall graphs of the NYTimes corpus are shown in Figure 4.2. In general, NYTimes recall curves follow the same trend as the WaPost corpus. Unlike the WaPost

experiments for relations P26P580, P569, P570 and P571, whose recall curves stop at the middle of the graph, all NYTimes experiments proceed until we review all 300,000 facts since there are enough $\langle Person,\ Year \rangle$ and $\langle Organization,\ Year \rangle$ entity pairs in the dataset.

Recalls @ $aN$ are manifested in Table 4.8 with mean recalls @ $3N, 6N, 12N, 24N$ being 22.80%, 34.05%, 47.51%, and 57.69%, correspondingly. By comparing with the WaPost results, it is interesting to observe that their mean recalls @ $aN$ for all $a \in \{3, 6, 12, 24\}$ are very close to each other with differences less than $\pm 1.50\%$, which proves the stability of our system across different collections of data.

| Relation | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| P22 father | 12.91 | 25.95 | 49.36 | 63.67 |
| P25 mother | 4.50 | 9.00 | 17.93 | 32.22 |
| P26 spouse | 14.69 | 35.40 | 54.85 | 61.29 |
| P3373 sibling | 5.65 | 14.78 | 33.71 | 45.69 |
| P40 child | 6.37 | 25.57 | 50.09 | 62.95 |
| P112 foundedBy | 32.43 | 45.56 | 55.08 | 62.71 |
| P169 CEO | 8.15 | 16.30 | 32.60 | 51.05 |
| P69 educatedAt | 27.92 | 45.72 | 64.52 | 76.96 |
| P26P580 yearOfMarriage | 12.19 | 24.24 | 35.85 | 39.66 |
| P569 yearOfBirth | 50.30 | 54.65 | 56.69 | 61.47 |
| P570 yearOfDeath | 56.19 | 61.23 | 63.56 | 66.89 |
| P571 yearFounded | 42.32 | 50.25 | 55.87 | 67.74 |

Table 4.8: Recall @ aN for the NYTimes corpus.

### 4.3.2 Ablation Experiments

To investigate the effectiveness of features, we conduct ablation experiments on the WaPost Corpus. As mentioned in Section 4.1.3, our original feature set consists of the entity positioning flag, TF-IDF scores, a union of unigram and bigram lexical features, and syntactic features, so we remove each of them (except the positioning flag) individually, denoted as -Tfidf, -ULex, and -Syn. Since we are also curious about whether bigram lexical features are essential to our feature set, we also run experiments without it and denote it as -BLex. Furthermore, we consider an additional feature introduced in Section 3.3.2, word vectors, and add it to our feature set, denoted as +WVec.

Final recalls for all feature settings are exhibited in Table 4.9, with the highest final recall for each relation being bold. Our original feature vector achieves the highest final

| Features | P22 | P25 | P26 | P3373 | P40 | P112 | P169 | P69 | P26P580 | P569 | P570 | P571 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Orig | **81.83** | 83.60 | **77.71** | **75.82** | 83.29 | 79.37 | 95.56 | 87.43 | **99.74** | **98.76** | **99.27** | **99.07** |
| -Tfidf | 81.35 | 83.92 | 77.50 | 74.78 | 83.03 | 79.37 | 95.56 | 87.47 | **99.74** | **98.76** | **99.27** | **99.07** |
| -Syn | 80.02 | 83.92 | **77.71** | 74.93 | **83.55** | 79.37 | 94.52 | 87.43 | **99.74** | **98.76** | **99.27** | **99.07** |
| -ULex | 80.51 | 83.28 | 77.64 | 75.07 | 82.68 | **79.72** | 95.82 | **87.49** | **99.74** | **98.76** | **99.27** | **99.07** |
| -BLex | 81.71 | 83.92 | 77.50 | 74.78 | 83.20 | **79.72** | **96.34** | 87.43 | **99.74** | **98.76** | **99.27** | **99.07** |
| +WVec | **81.83** | **84.57** | 77.43 | 74.63 | 83.20 | 79.37 | 95.82 | 87.43 | **99.74** | 98.26 | 99.04 | 98.23 |

Table 4.9: Final recalls of 12 relations from the WaPost corpus. Features are removed or added alternatively to conduct ablation experiments.

| Features | P22 | P25 | P26 | P3373 | P40 | P112 | P169 | P69 | P26P580 | P569 | P570 | P571 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Orig. | 221.55 | 226.56 | 208.30 | 195.22 | 226.94 | 214.36 | 270.57 | 224.47 | 202.72 | 209.97 | 193.70 | 112.43 |
| -Tfidf | 221.78 | 227.07 | **207.79** | 194.25 | 226.06 | 214.19 | 270.70 | 224.33 | 202.58 | 209.94 | 193.83 | 112.34 |
| -Syn | **219.62** | 225.98 | 208.12 | 194.47 | **224.14** | **213.14** | 268.52 | 222.55 | **202.45** | 210.01 | 193.81 | 112.33 |
| -ULex | **218.99** | **225.64** | 207.84 | **193.98** | **220.80** | 214.10 | **264.66** | **221.48** | 202.62 | **209.77** | **193.33** | **112.10** |
| -BLex | 224.07 | 226.49 | 208.13 | 194.27 | 224.33 | 216.30 | 270.75 | 224.37 | 202.54 | 209.97 | 193.72 | 112.44 |
| +WVec | 222.00 | **223.37** | **206.51** | **194.07** | 224.53 | **213.00** | **266.87** | **219.65** | **202.42** | **201.52** | **192.73** | **109.26** |

Table 4.10: Areas under recall curves of 12 relations from the WaPost corpus. Features are removed or added alternatively to conduct ablation experiments.

recall among all settings at the most number of times, but there is no obvious difference between our original feature group's final recalls and other feature groups'. The impact of each feature varies among different relations. For example, excluding TF-IDF reduces final recalls for relation *P22*, *P26*, *P3373*, and *P40*, but it improves the results for relations *P25* and *P69*. Additionally, it is interesting that removing the bigram lexical features achieves comparable results as the original feature sets. Three relations reach higher recalls than the original setting, but four other relations terminate at a lower final recall, which means removing bigram lexical features from the feature set does not influence our system's effectiveness much. In conclusion, the choice of features used to train the classifier does not notably affect our final recall results, but adding word vectors (i.e., +WVec) slows the system run time by more than five times.

Figure 4.3 shows recall graphs for all feature settings, where curves in different colours are generated using different feature groups. We observe that the general shape of curves are similar, and curves sometimes overlap each other. In graphs of relations *P22*, *P25*, and *P169*, the +WVec and -ULex curves are below other curves at their knee areas, which means that when most of "obvious" positive facts are identified, it will be harder for the classifier trained by +WVec or -ULex feature groups to recognize positive facts than the classifier trained by other features. However, +WVec and -ULex curves catch up with other curves at a later point, indicating that the classifier trained by +WVec and -ULex

feature groups can identify positive fact as other trained classifiers, but it would need to evaluate more facts. In other words, if the enterprise is willing to evaluate enough facts, there is no much difference between training the classifier using different features, but if the enterprise decides to terminate at curves' knee points, then systems with +WVec and -ULex settings might find fewer positive facts than other feature settings. Furthermore, as manifested in Table 4.10, areas under the recall curves are calculated for all feature groups, and we mark the two smallest areas as bold numbers for each relation. As we can expect from their recall graphs, +WVec and -ULex settings frequently result in the two smallest areas under their curves among all feature settings.

Different feature groups' recalls @ $aN$ are shown in Appendix B. Recalls achieved using different features are generally close to each other. Overall, the original group and -BLex group most often produce the highest recall among all groups. When $a \in \{6, 12, 24\}$, the original group has the highest recall most times; when $a = 3$, the original group achieves the highest recall for five relations while the -BLex group achieves the highest recall for six relations. This is another evidence that the bigram lexical features do not play an important role in improving the system's effectiveness. Moreover, since +WVec and -Syn groups rarely achieve higher recalls than the original group, we can conclude that adding word vectors usually damages the system's effectiveness, and it is vital to have syntactic features to boost the recall.

In summary, different feature groups have achieved similar results in all evaluation metrics, with the original feature group obtaining slightly higher final recalls and recalls @ $aN$. Based on our observations from recall graphs and recalls @ $aN$, unigram lexical features and syntactic features help identify positive relation instances before negative ones. Besides, evaluation results with and without the bigram lexical features are always very close to each other. Last but not least, adding word vectors hurts the system's effectiveness for most relations and makes our system's run time slower by more than five times.

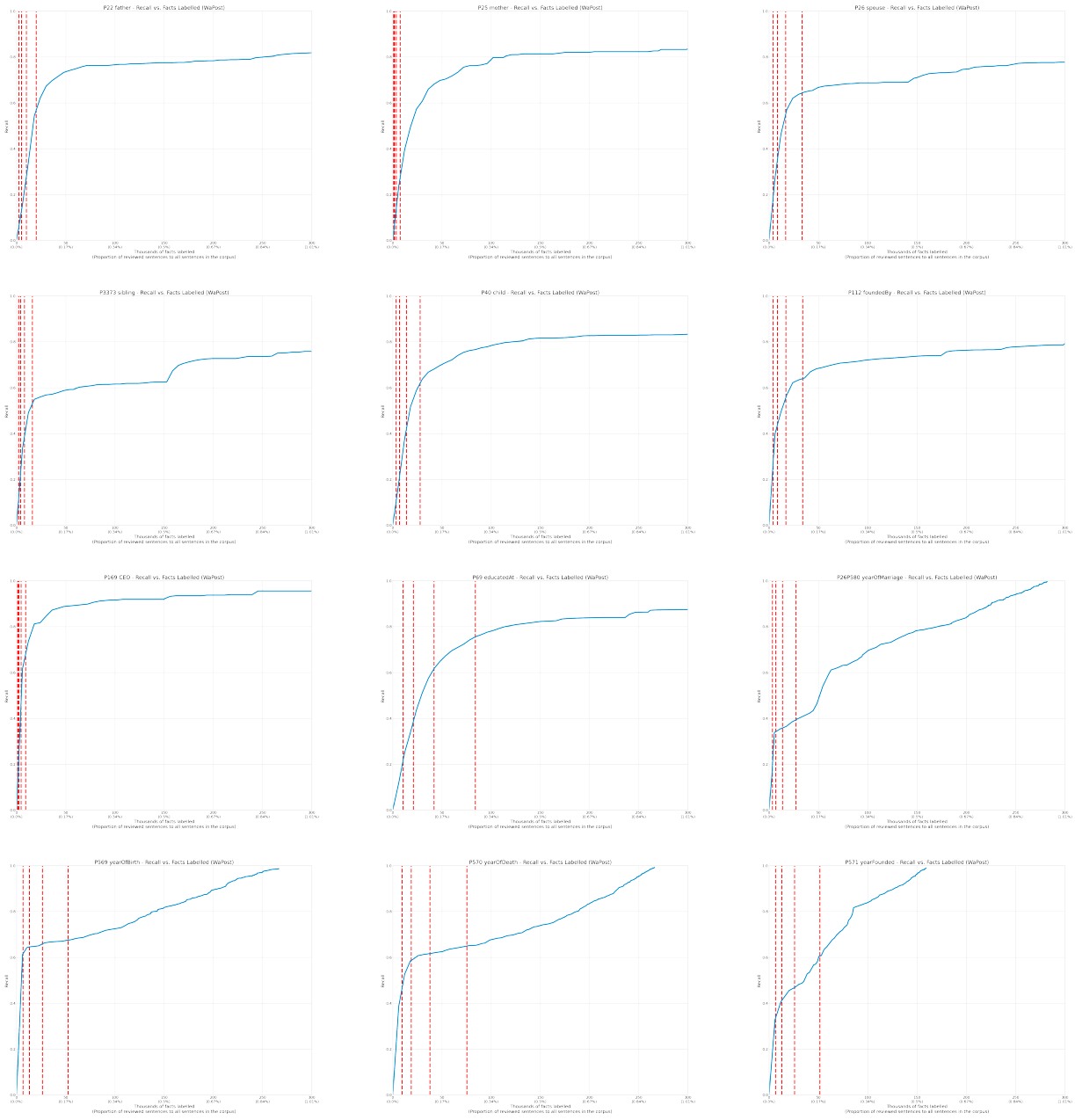


Figure 4.1: Recall curves of 12 relations from the WaPost corpus.

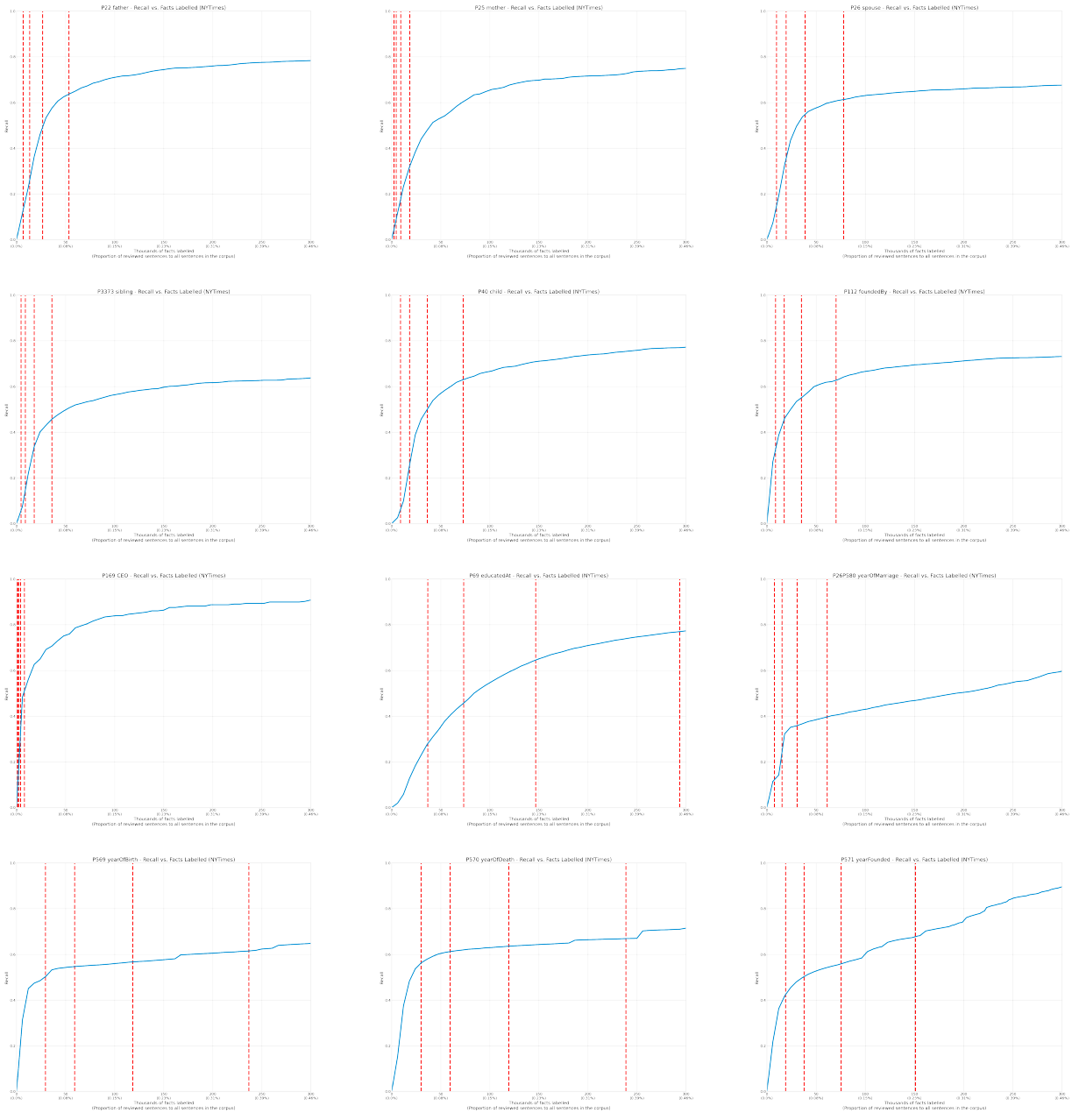


Figure 4.2: Recall curves of 12 relations from the NYTimes corpus.

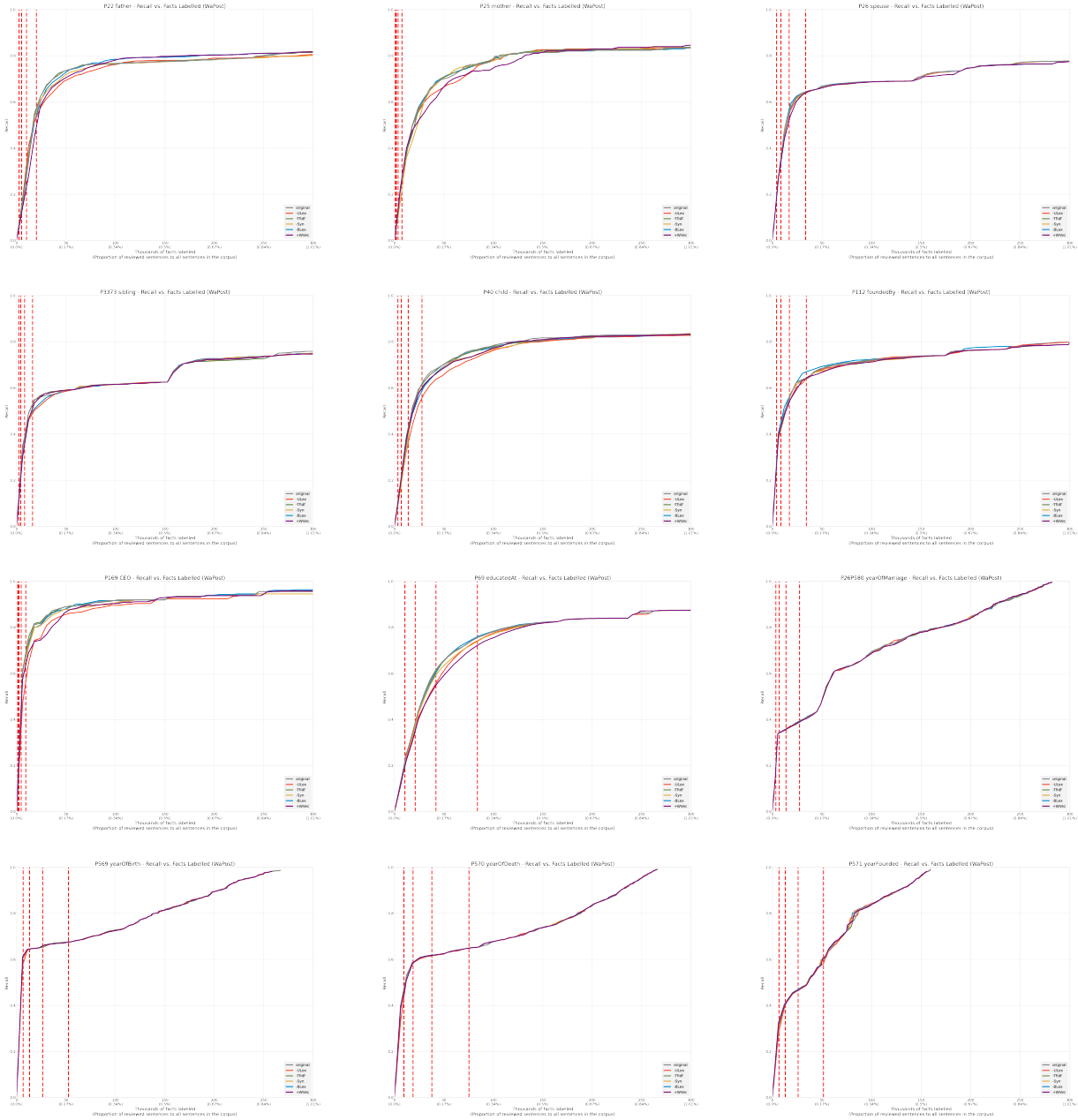


Figure 4.3: Recall curves of 12 relations from the WaPost corpus. Features are removed or added alternatively to conduct ablation experiments, denoted as -Tfidf, -Syn, -ULex, -BLex, and +WVec.

# Chapter 5

# Conclusion and Future Work

In this thesis, we propose a novel task, Total Relation Recall (TRR), to help enterprises create Knowledge Graphs using their unstructured data. Given a collection of documents and a enterprise-specified relation, TRR wants to find nearly all or nearly all facts of the specified relation from the given dataset, where human assessors are involved in this process to annotate each extracted relation facts.

We present a Python-based system providing a baseline method to the TRR problem, which is inspired and implemented based on the Baseline Model Implementation (BMI) in the TREC Total Recall Track. Our system works in an iterative style, with each iteration comprises three major components: Retrieval, Classification, and Feedback Modules. The Retrieval Module discovers candidate entity pairs with contexts potentially expressing the target relation. Then the Classification Module utilizes the logistic regression classifier to rank all discovered candidate entity pairs based on their likelihoods of being the target relation's instances, and top $b$ candidate are passed to human assessors as a batch of candidate facts, where $b$ is the number of facts that reviewers label in each iteration. The Feedback Module takes the assessment results as input and outputs all positive facts in the results, revises the logistic regression classifier by expanding its training set with labelled facts, and explores words frequently appearing in positive facts, which will be used for the Retrieval Module in the next iteration.

Moreover, we evaluate the effectiveness of our system using the simulation study. Our experiments are conducted with two corpora of news articles and 12 relations of four different entity types, with human assessors simulated by the commonly used Knowledge Base (KB) Wikidata. The results show that our system can achieve relatively high recalls with reasonable small assessment effort for all experimental settings. We also conduct an

ablation study on different NLP features used in training the classifier, but no feature consistently improves the recall over all relations.

In future work, we would like to investigate how the target relation's information would impact our system's effectiveness, including a different set of relation keywords and different initial instances. Similarly, it is worth exploring different classifiers as well. Our system uses the logistic regression classifier to rank candidate entity pairs with contexts, but the logistic regression can be switched to any other types of classifiers easily, such as the Support Vector Machine (SVM) or Light Gradient Boosting Method (LightGBM).

Furthermore, as discussed in Section 4.1.2, our simulated human assessments may produce false positives and false negatives, which may directly affect the quality of our classifier and explored relation keywords. Likewise, all relations' ground truth is automatically generated using the Wikidata knowledge, and the number of our ground truth is different from the actual number of relation instances in the dataset because of the NER/entity linker error and false positive/negative problem (see Section 4.2), which could influence the recall calculated in our experiments. Therefore, it would be helpful to manually label all "gold" ground truth and run experiments with human reviewers.

Finally, we would like to evaluate our system with user-defined relations that do not exist in any KBs. In our experiments, the 12 relations used are chosen from Wikidata. Whereas, thinking of relations in real life, a considerable number of them does not exist in any KBs, such as $\langle Person,\ favoriteVacationPlace,\ Location \rangle$. Our system theoretically works with any well-defined relations regardless of their existences in KBs. However, the most challenging part is that in our experimental setup, the ground truth of non-KB relations cannot be generated using Wikidata information, and the human assessor cannot be simulated either. As the manual evaluation is costly, we leave experiments with non-KB relations for future research.

# References


[1] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[2] G. Cormack and M. R. Grossman. Waterloo (cormack) participation in the trec 2015 total recall track. In *TREC*, 2015.

[3] Gordon V. Cormack and Maura R. Grossman. Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. In *Proceedings of the 37th International ACM SIGIR Conference on Research amp; Development in Information Retrieval*, SIGIR '14, page 153–162, New York, NY, USA, 2014. Association for Computing Machinery.

[4] Gordon V. Cormack and Maura R. Grossman. Autonomy and reliability of continuous active learning for technology-assisted review, 2015.

[5] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, December 2008.

[6] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 1156–1165, New York, NY, USA, 2014. Association for Computing Machinery.

[7] M. R. Grossman, G. Cormack, and Adam Roegiest. Trec 2016 total recall track overview. In *TREC*, 2016.

[8] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength natural language processing in python, 2020.

[9] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: An easy-to-use python toolkit to support replicable ir research with sparse and dense representations, 2021.

[10] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany, August 2016. Association for Computational Linguistics.

[11] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[12] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[13] Makoto Miwa and Mohit Bansal. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany, August 2016. Association for Computational Linguistics.

[14] Thien Huu Nguyen and Ralph Grishman. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48, Denver, Colorado, June 2015. Association for Computational Linguistics.

[15] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In José Luis Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 148–163, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[16] Adam Roegiest and Gordon V Cormack. Total recall track tools architecture overview. *Proc. TREC-2015*, 2015.

[17] Adam Roegiest, Gordon V Cormack, Charles LA Clarke, and Maura R Grossman. Trec 2015 total recall track overview. In *TREC*, 2015.

[18] Christopher C. Shilakes and Julie Tylman. Enterprise information portals. *Merrill Lynch*, 1998.

[19] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 304–311, New York City, USA, June 2006. Association for Computational Linguistics.

[20] Alisa Smirnova and Philippe Cudré-Mauroux. Relation extraction using distant supervision: A survey. *ACM Comput. Surv.*, 51(5), November 2018.

[21] Johannes M. van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P. de Vries. Rel: An entity linker standing on the shoulders of giants. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 2197–2200, New York, NY, USA, 2020. Association for Computing Machinery.

[22] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, September 2014.

[23] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 1253–1256, New York, NY, USA, 2017. Association for Computing Machinery.

[24] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Reproducible ranking baselines using lucene. *J. Data and Information Quality*, 10(4), October 2018.

[25] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 353–362, New York, NY, USA, 2016. Association for Computing Machinery.

[26] Xiaohan Zou. A survey on application of knowledge graph. *Journal of Physics: Conference Series*, 1487:012016, March 2020.

# APPENDICES

# Appendix A

# Initial Instances in Experiments

For each of the 12 relations evaluated in Chapter 4, 10 relation instances are extracted from the TREC Washington Post Corpus[1] and The New York Times Annotated Corpus[2] as system input.

## A.1 TREC Washington Post Corpus

### P22 father

⟨**Ivanka Trump, Donald Trump**⟩: "Friends are good, but family is better," **Trump** wrote, foreshadowing his heavy reliance during the campaign on his children, particularly his daughter **Ivanka** and her husband, Jared Kushner.

⟨**Lachlan Murdoch, Rupert Murdoch**⟩: Ailes's sins were worse, and seemed to be more pervasive; on his way out the door, **Lachlan** and James Murdoch alluded to the circumstances behind his departure, though they joined their father, **Rupert**, in praising his vision.

⟨**Ravi Coltrane, John Coltrane**⟩: **Ravi Coltrane** was nearly 2 years old when his father, jazz legend **John Coltrane**, died.

⟨**Kyle Shanahan, Mike Shanahan**⟩: When asked Monday about the possibility of **Kyle Shanahan**, his son, being fined by the league, **Mike Shanahan** said: "I think we'll let

[1] https://trec.nist.gov/data/wapost/
[2] https://catalog.ldc.upenn.edu/LDC2008T19

the powers that be take a look at what transpired on both sides and let them make the decision."

⟨**Rand Paul, Ron Paul**⟩: Mila Farrell, 50, volunteered for all three of former Texas congressman **Ron Paul**'s campaigns, and she drove 6 1/2 hours to help his son **Rand Paul** open the Las Vegas office.

⟨**John T. Walton, Sam Walton**⟩: Her husband, **John Walton**, was one of Wal-Mart founder **Sam Walton**'s four children.

⟨**Gloria DeHaven, Carter DeHaven**⟩: **Gloria Mildred DeHaven** was born in Los Angeles on July 23, 1925, the youngest of three children of **Carter DeHaven** and the former Flora Parker.

⟨**Prince Andrew, Prince Philip**⟩: The queen and **Prince Philip** stand with their children, from left, Prince Charles, 17; Princess Anne, 15 and **Prince Andrew**, 5.

⟨**Lorin Maazel, Lincoln Maazel**⟩: **Lorin Varencove Maazel** was born March 5, 1930, in Neuilly-sur-Seine, France, near where his parents — singer-actor **Lincoln Maazel** and pianist Marie Varencove — were studying music in Paris.

⟨**Allison Williams, Brian Williams**⟩: **Allison Williams** defends dad **Brian Williams** after NBC suspension: 'I know you can trust him'

**P25 mother**

⟨**Katharine Weymouth, Lally Weymouth**⟩: And Trump had a fraught relationship with the WHCA dinner going back to 2011, when he was the highly controversial guest of **Lally Weymouth**, mother of then-Washington Post publisher **Katharine Weymouth**.

⟨**Jeb Bush, Barbara Bush**⟩: **Jeb Bush** is scheduled to appear with his mother **Barbara Bush** at a campaign event in Derry, N.H.

⟨**Charles(Prince of Wales), Elizabeth II**⟩: This photo, circa 19951, shows the royal British couple, **Queen Elizabeth II**, and her husband Philip, Duke of Edinburgh, with their children, **Charles**, Prince of Wales, and Princess Anne.

⟨**Chelsea Clinton, Hillary Clinton**⟩: **Hillary Clinton** accompanied by her husband, former President Bill Clinton, and daughter, **Chelsea Clinton**.

⟨**Bobbi Kristina, Whitney Houston**⟩: **Bobbi Kristina**, 22, is the only child of **Whitney Houston** and Bobbi Brown.

⟨**Kim Jong-chul, Ko Yong-hui**⟩: He then had at least one more relationship, with a Japanese-born Korean, **Ko Young Hee**, who gave birth to two more children — **Kim Jong Chul** and Kim Jong Eun.

⟨**John F. Kennedy Jr., Jacqueline Kennedy Onassis**⟩: After her husband's assassination, **Jacqueline Kennedy** and her children, Caroline and **John Jr.**, returned to the neighborhood, living at 3038 N St.

⟨**Kim Kardashian, Kris Jenner**⟩: In an interview with **Kris Jenner**, mother of his girlfriend **Kim Kardashian**, that aired today, West was mostly there to show off a photo of North West, his daughter with Kardashian, and chat—somewhat uncomfortably—about living in Jenner's house.

⟨**Kim Kardashian, Kris Jenner**⟩: **Kim Kardashian**, right, poses with her mother, **Kris Jenner**, in 2009.

⟨**Bran Stark, Catelyn Stark**⟩: In that scene, an unknown assassin fought off **Bran**'s mom, **Catelyn Stark**, but Bran's direwolf kills the guy.

**P26 spouse**

⟨**Bill Clinton, Hillary Clinton**⟩: Chelsea Clinton joined her mother, Democratic presidential candidate **Hillary Clinton**, and father, former president **Bill Clinton**, on caucus night in Iowa on Monday.

⟨**Bill Clinton, Hillary Clinton**⟩: In this July 21, 1992 file photo, then-Democratic presidential nominee **Bill Clinton** stands with his wife **Hillary Clinton** during a campaign stop.

⟨**Newt Gingrich, Callista Gingrichs**⟩: In late October, **Gingrich** and his wife, **Callista**, attended the ribbon-cutting event for the property.

⟨**Newt Gingrich, Callista Gingrichs**⟩: Marianne Gingrich, to whom **Gingrich** was married when he began an affair with his current wife, **Callista**, had been in the news all day as excerpts of a then soon-to-be-released interview with ABC News were replayed dozens of times.

⟨**Jared Kushner, Ivanka Trump**⟩: Last weekend, he dined there with his wife, Melania; previously, he had eaten there with his daughter **Ivanka** and her husband, **Jared Kushner**, a senior adviser.

⟨**Jared Kushner, Ivanka Trump**⟩: **Ivanka Trump** and **Jared Kushner** — now both senior White House aides — were married there in 2009.

⟨**Michelle Obama, Barack Obama**⟩: President **Barack Obama** and first lady **Michelle Obama** acknowledge the crowd as they arrive in the 440th Structural Maintenance Hangar at Fort Bragg, N.C., Wednesday, Dec.

⟨**Michelle Obama, Barack Obama**⟩: "In these girls, **Barack** and I see our own daughters," **Michelle Obama** said in a five-minute address last weekend.

⟨**Melania Trump, Donald Trump**⟩: Clinton and former President Bill Clinton were photographed at the event laughing and talking with **Trump** and wife **Melania Knauss**.

⟨**Melania Trump, Donald Trump**⟩: President **Donald Trump** waves as he walks with first lady **Melania Trump** during the inauguration parade on Jan.

**P3373 sibling**

⟨**Vitaliy Klitschko, Volodymyr Klitschko**⟩: The Klitschko era soon followed — in which **Wladimir** and his brother, **Vitali**, were thoroughly dominant.

⟨**James Murdoch, Lachlan Murdoch**⟩: Instead, Shine appeared to come under increasing pressure all week, as rumors began circulating that Murdoch's sons — **Lachlan** and **James**, who run Fox's parent company, 21st Century Fox — were seeking his successor.

⟨**James Murdoch, Lachlan Murdoch**⟩: **Lachlan Murdoch**, left, his father Rupert and brother **James** at Rupert's marriage to former model Jerry hall in London in 2016.

⟨**Sonny Corleone, Michael Corleone**⟩: In "The Godfather," **Michael Corleone** became a gangster after his brother **Sonny** was brutally slaughtered on the causeway in a dispute over drugs.

⟨**Alaa Mubarak, Gamal Mubarak**⟩: **Alaa** and **Gamal Mubarak**, the former president's two sons, are in prison facing corruption charges.

⟨**Alaa Mubarak, Gamal Mubarak**⟩: His sons, **Alaa** and **Gamal**, also were charged with embezzling millions through a network of official cronyism.

⟨**Jim Harbaugh, John Harbaugh**⟩: He played for **Jim Harbaugh**, the brother of Ravens Coach **John Harbaugh**, in San Francisco.

⟨**David Koch, Charles Koch**⟩: The group spent more than $100 million on politics last year, partly with support from billionaire brothers **Charles** and **David Koch**, owners of a Kansas-based energy and chemical conglomerate.

⟨**David Koch, Charles Koch**⟩: Two of Bill Koch's brothers, **Charles** and **David**, lead the influential Koch political network, which has refused to get behind Trump's campaign.

⟨**Venus Williams, Serena Williams**⟩: **Venus Williams** admitted that she always has an escape plan and her sister, **Serena**, was stalked by a man who tracked her down via social media.

**P40 child**

⟨**Lally Weymouth, Katharine Weymouth**⟩: And Trump had a fraught relationship with the WHCA dinner going back to 2011, when he was the highly controversial guest of **Lally Weymouth**, mother of then-Washington Post publisher **Katharine Weymouth**.

⟨**Donald Trump, Ivanka Trump**⟩: **Trump** and his daughter **Ivanka** by no longer shopping at Nordstrom and Neiman Marcus.

⟨**Dick Cheney, Elizabeth Cheney** ⟩: Reid was responding to **Cheney**'s op-ed in the Wall Street Journal with his daughter **Liz** attacking the Obama administration's policies in the Middle East and elsewhere, a piece that has already generated much discussion.

⟨**Hillary Clinton, Chelsea Clinton**⟩: **Hillary Clinton**, with daughter **Chelsea Clinton**, reacts as President Barack Obama speaks at the Clinton Global Initiative in New York City in September.

⟨**Donald Trump, Donald Trump Jr.**⟩: **Trump** has largely resisted calls by ethics experts for full divestiture, saying his sons **Don Jr**.

⟨**Donald Trump, Donald Trump Jr.**⟩: **Trump**'s son, Donald **Trump Jr.**, has also campaigned in Montana for Gianforte.

⟨**Jill Biden, Beau Biden**⟩: **Jill Biden**, center, wife of Vice President Biden, sits with her sons **Beau Biden**, left, and Hunter Biden, right, before the start of the vice presidential debate, at Centre College in Danville, Ky., Thursday, Oct.

⟨**Lord Byron, Ada Lovelace**⟩: **Lovelace**, who died in 1852, was the only legitimate child of the poet **Lord Byron**.

⟨**Mike Huckabee, Sarah Sanders**⟩: "It's easier to be mad at CNBC," said **Sarah Huckabee Sanders**, the campaign manager for her father, **Mike Huckabee**.

⟨**Lincoln Maazel, Lorin Maazel**⟩: **Lorin Varencove Maazel** was born March 5, 1930, in Neuilly-sur-Seine, France, near where his parents — singer-actor **Lincoln Maazel** and pianist Marie Varencove — were studying music in Paris.

**P112 foundedBy**

⟨**Amazon, Jeff Bezos**⟩: The Post is owned by **Amazon** founder and chief executive **Jeffrey P. Bezos**.

⟨**Microsoft, Bill Gates**⟩: **Bill Gates** is the founder of **Microsoft**.

⟨**The Princeton Review, John Katzman**⟩: Co-founder **John Katzman**, formerly of **The Princeton Review**, will remain with the company as executive chairman.

⟨**Forstmann Little & Company, Theodore J. Forstmann**⟩: These men — Henry Kravis and his cousin George Roberts, the founders of KKR & Co.; the late **Teddy Forstmann**, the founder of **Forstmann Little**; David Bonderman and Jim Coulter, the founders of TPG Capital; Leon Black, the founder of Apollo Global Management; Steve Schwarzman and Pete Peterson, the founders of the Blackstone Group; David Rubenstein, the founder of the Carlyle Group; and Jonathan Nelson, the founder of Providence Equity Partners — each have a net worth measured in the billions.

⟨**Ares Management, Antony Ressler**⟩: Another group of potential bidders includes three people who have ties to major league sports: **Tony Ressler**, the Los Angeles-based co-founder of the investment firm **Ares Management**, who holds a minority stake in the Milwaukee Brewers; Oaktree Capital Management co-founder Bruce Karsh, who is a minority owner of the NBA's Golden State Warriors and chairman of Los Angeles Times owner Tribune Co.; and Grant Hill, a onetime NBA all-star who finished his career with the Clippers.

⟨**Palantir Technologies, Alex Karp**⟩: A considerable conflict of interest, wrote Sullivan, had poisoned a recent T story anointing a group of "Five Visionary Tech Entrepreneurs Who Are Changing the World." Titled "The Transformers" in print, the piece boiled down to five discrete puff pieces on Elizabeth Holmes (chief executive of blood test company Theranos), Vineet Singal (co-founder and chief executive of CareMessage), Brian Chesky (co-founder and chief executive of Airbnb), Leila Janah (founder of the Sama Group) and **Alex Karp** (co-founder and chief executive of **Palantir Technologies**).

⟨**Oracle Corporation, Larry Ellison**⟩: "Moore's decision was announced in a statement by **Larry Ellison**, the billionaire co-founder of **Oracle Corp.** and owner of the tournament.", Oracle Corp.

⟨**Human Longevity, Craig Venter**⟩: **Craig Venter**, co-founder of **Human Longevity Inc.**; Martine Rothblatt, co-chief executive of United Therapeutics Corp.; Arati Prabhakar, director of DARPA; NASA Administrator Charles Bolden; Emmett Shear, chief executive of Twitch; David Rubenstein, co-founder of the Carlyle Group; Wendy Schmidt, president,

The Schmidt Family Foundation and co-founder, Schmidt Ocean Institute; George Whitesides, chief executive of Virgin Galactic; Helen Greiner, chief executive of CyPhy Works; Steve Huffman, co-founder of Reddit; David Kenny, general manager, IBM Watson; Neil Harbisson, cyborg artist; Katie Couric, journalist, author and Yahoo Global News Anchor; and Martin Baron, executive editor of The Washington Post.

⟨**Qualcomm Inc., Irwin M. Jacobs**⟩: Other technology leaders on the list included eBay founder Pierre Omidyar and his wife, Pam, **Qualcomm** founder **Irwin Jacobs** and his wife, Joan, Google co-founder Sergey Brin and his estranged wife, Anne Wojcick, and Microsoft co-founder Paul Allen.

⟨**Blackboard Inc., Michael Chasen**⟩: The company's financial backers include New Enterprise Associates, **Blackboard** and Social Radar founder **Michael Chasen**, Twitter co-founder Evan Williams, and Amazon.com founder and Washington Post owner Jeffrey P. Bezos.

**P169 CEO**

⟨**Koch Industries, Charles Koch**⟩: **Charles Koch** is chairman and chief executive of **Koch Industries**.

⟨**Amazon, Jeff Bezos**⟩: Disclosure: **Amazon** chief executive **Jeff Bezos** owns The Washington Post.

⟨**Apple Inc., Tim Cook**⟩: CEO **Tim Cook** holds up the billionth iPhone sold in front of employees at **Apple** headquarters in July 2016.

⟨**Blackstone, Stephen A. Schwarzman**⟩: **Schwarzman**, **Blackstone**'s chairman and chief executive, is close to Trump and leads the White House's economic advisory council of CEOs.

⟨**Tesla, Elon Musk**⟩: Trump met Monday with business leaders from a smattering of industries, including Fields and **Tesla** chief executive **Elon Musk**.

⟨**Yahoo, Scott Thompson**⟩: **Yahoo** named a new chief executive Wednesday morning: **Scott Thompson**, most recently known as president of PayPal.

⟨**Yahoo, Carol Bartz**⟩: In June 2011, **Yahoo** chairman Roy Bostock praised the company's first female chief executive, **Carol Bartz**, at the annual shareholder meeting.

⟨**US Airways, Doug Parker**⟩: **US Airways** chief executive **Doug Parker** received a 44 percent increase in compensation, to $5.5 million, last year.

⟨**Microsoft, Steve Ballmer**⟩: **Microsoft** CEO **Steve Ballmer** speaks at the 2011 International Consumer Electronics Show.

⟨**Intel, Paul Otellini**⟩: The afternoon brings yet another keynote speech, this time from **Intel** president and chief executive **Paul Otellini**.

**P69 educatedAt**

⟨**University of Maryland, Len Bias**⟩: Ramirez (D-Prince George's) to obtain state funds to place a statue of **Len Bias**, a **University of Maryland** basketball star, at Northwestern High School in Hyattsville.

⟨**University of Virginia, Otto Warmbier**⟩: Two people — **University of Virginia** student **Otto Warmbier** and Korean American missionary Kim Dong-chul — are currently being held.

⟨**University of Virginia, Charmaine Yoest**⟩: **Yoest**, a breast cancer survivor and mother of five who lives in Virginia, holds a PhD in government from the **University of Virginia**.

⟨**University of Maryland, Julian Ivey**⟩: It was the first run for political office for **Julian Ivey**, a junior at the **University of Maryland** who served as a delegate for Sen.

⟨**University of Virginia, Elizabeth Haysom**⟩: At the time of the murders, he and his girlfriend, **Elizabeth Haysom**, had been honors students at the **University of Virginia**.

⟨**Phillips Exeter Academy, Mark Zuckerberg**⟩: Emily Talmage is an elementary school teacher in Lewiston, Maine, who happened to be a classmate of Facebook founder **Mark Zuckerberg** when they both attended **Phillips Exeter Academy** as teenagers.

⟨**The Catholic University of America, Terry McAuliffe** ⟩: **McAuliffe**, 53, earned a BA in political science from **Catholic University** and a law degree at Georgetown University before practicing banking and securities law for several years, according to the first lady's official biography.

⟨**Harvard University, Matthew Yglesias**⟩: Besides Sullivan, who has a PhD in political philosophy and is known for his writings on conservatism and gay marriage, the other participants included Slate blogger **Matthew Yglesias**, who majored in philosophy at **Harvard**, and Grist magazine writer/blogger David Roberts, who has a master's degree in philosophy from the University of Montana."

⟨**Yale law school, Bill Clintons**⟩: **Clinton** was just out of **Yale law school** when he ran his first campaign, for Arkansas' 3rd district in the U.S.

⟨**Wellesley College, Barbara Lea**⟩: She graduated from **Wellesley College** in Massachusetts in 1951 with a bachelor's degree in music theory and began singing in clubs under the name **Barbara Lea**.

**P26P580 yearOfMarriage**

⟨**Jared Kushner, 2009**⟩: Ivanka Trump and **Jared Kushner** — now both senior White House aides — were married there in **2009**.

⟨**Jared Kushner, 2009**⟩: His daughter Ivanka Trump underwent an Orthodox conversion before her **2009** marriage to **Jared Kushner**, who was raised observant.

⟨**Tom Brady, 2009**⟩: Two bodyguards who shot at paparazzi at the **2009** wedding of Gisele Bundchen and **Tom Brady** in Costa Rica were convicted of attempted murder and sentenced to five years in prison.

⟨**Barack Obama, 1992**⟩: When **Barack** and Michelle Obama married in **1992**, they reflected the evolution of marriage in the United States.

⟨**Sarah Ferguson, 1986**⟩: Andrew married **Sarah Ferguson**, nicknamed "Fergie," in **1986**, but they separated not long after.

⟨**Prince Andrew, 1986**⟩: The ambassador held elaborate dinners for heads of state at his London home, and hosted the American delegation attending the **1986** royal wedding of **Prince Andrew** and Sarah Ferguson.

⟨**Gisele Bundchen, 2009**⟩: Two bodyguards who shot at paparazzi at the **2009** wedding of **Gisele Bundchen** and Tom Brady in Costa Rica were convicted of attempted murder and sentenced to five years in prison.

⟨**LeAnn Rimes, 2011**⟩: Rimes and **Cibrian** wed in **2011** — and ever since then, Rimes and Glanville have been publicly bickering on Twitter.

⟨**Margot Honecker, 1953**⟩: Party leaders asked his wife to grant him a divorce "as a patriotic duty." Erich and **Margot Honecker** were married in **1953**, one year after their daughter was born.

⟨**Ross Kemp, 2002**⟩: Brooks married her first husband, **Ross Kemp**, in **2002**.

**P569 yearOfBirth**

⟨**Joe Biden, 1942**⟩: As the article points out, **Biden** is 70 — born in **1942**.

⟨**Donald Trump, 1946**⟩: Though birthers may falsely claim that President Obama was born in Kenya, the nation can, one would hope, agree that **Donald Trump**, 70, was born in Queens in **1946** and that Hillary Clinton, 68, was born in Chicago in 1947.

⟨**John McCain, 1936**⟩: **McCain** was born in the Panama Canal Zone in **1936**, and the Senate unanimously passed a resolution declaring him a natural born citizen when he ran for president in 2008.

⟨**Roger Straus, 1917**⟩: **Roger Straus** (**1917**-2004), the scapegrace scion of one branch of the Guggenheim family, founded the publishing house that bears his name in 1945.

⟨**Ben Bradlee, 1921**⟩: In memory of **Ben Bradlee**, **1921-2014**, here's the beginning of a lecture he gave on public dishonesty: Ï would like to talk about government lying.

⟨**Barbara Cushing, 1915**⟩: Born **Barbara Cushing** in **1915**, she was raised to marry well by an ambitious mother, a goal she achieved brilliantly when she became the second wife of William Paley, the megalomaniacal founder of CBS.

⟨**Edmund Dulac, 1882**⟩: Hughey, who was trained as a lawyer, later developed an interest in the art of **Dulac**, who lived from **1882** to 1953.

⟨**Ashraf Pahlavi, 1919**⟩: **Ashraf Pahlavi** was born in Tehran on Oct. 26, **1919**, five hours after her twin brother — children of a military commander, Reza Pahlavi, and the second of his four wives.

⟨**Herman Wouk, 1915**⟩: To this company of long-lived legends, one should add novelist **Herman Wouk** (born in **1915**).

⟨**Balanchine, 1904**⟩: Preparation of a book-length biography of **Balanchine** (**1904**-1983), from his earliest years in Imperial Russia to his death in New York.

**P570 yearOfDeath**

⟨**Kathryn Steinle, 2015**⟩: Vaughan said the Cornyn- McCaul measure combines separate proposals that perhaps could pass on their own, including a bill known as "Kate's Law," named for **Kathryn Steinle**, who was shot and killed in **2015** in San Francisco, allegedly by a Mexican national who had been deported several times but had returned to the United States.

⟨**Nelson Mandela, 2013**⟩: It is was the fourth meeting between the leaders, who first shook hands at the **2013** funeral of **Nelson Mandela** in South Africa.

⟨**Jane Austen, 1817**⟩: Ever since **Jane Austen** died in **1817**, at 41, her novels have inspired the most intense veneration.

⟨**Princess Diana, 1997**⟩: Media commentators say that the post-Leveson world has potentially made the industry more cautions, but they also note that the media has been keen not to offend the palace in recent years, especially following the death of **Princess Diana** who died in a car crash in Paris in **1997** after being chased by paparazzi.

⟨**Antonin Scalia, 2016**⟩: But immediately after the unexpected death of Justice **Antonin Scalia** in February **2016**, McConnell said the Republican-controlled Senate would leave the job open so the next president could fill the vacancy.

⟨**Roh Moo-hyun, 2009**⟩: **Roh**, who committed suicide in **2009**, had a "philosophy of balanced national development," said a former prime minister, Lee Hae-chan.

⟨**Megan Kanka, 1994**⟩: It was named for 7-year-old **Megan Kanka** of Hamilton Township, N.J., who was raped and killed in **1994** by Jesse Timmendequas, a convicted child molester who was living across the street from the Kankas.

⟨**John Paul, 2005**⟩: Pope **John Paul**, who led the church from 1978 until his death in **2005**, had high popularity numbers among Americans.

⟨**Benazir Bhutto, 2007**⟩: **Bhutto** was killed in **2007** during a gun and bomb attack at a rally in Rawalpindi, the sister city of the Pakistani capital, Islamabad.

⟨**Mark Rothko, 1970**⟩: It is now the central repository for the study of **Rothko**, the Russian-born painter who lived from 1903 to **1970**.

### P571 yearFounded

⟨**National Football League, 1920**⟩: The **NFL** was founded in **1920**.

⟨**Waffle House, 1955**⟩: They opened the first 24-hour **Waffle House** in the Atlanta suburb of Avondale Estates on Labor Day in **1955**.

⟨**YouTube, 2005**⟩: It was **2005**, the same year that **YouTube** was founded and the idea of a "viral video" online was just entering the lexicon."

⟨**National Review, 1955**⟩: Since its founding in **1955** by William F. Buckley, **National Review** has always been an opinion journal.

⟨**World Trade Organization, 1995**⟩: It would be the largest international trade agreement since the **World Trade Organization** was created in **1995**.

⟨**World Trade Organization, 1995**⟩: The establishment of the **WTO** in **1995** ushered in a major new system for countries to resolve their bilateral trade frictions.

⟨**El Universo, 1921**⟩: Chavez, Venezuela's president, has taken over most of the television and radio media in his country, even he has not dared to attack historic newspapers like **El Universo**, which was founded in **1921** in Ecuador's coastal business capital, Guayaquil, and is widely respected across the region.

⟨**Facebook, 2004**⟩: Mark Zuckerberg was one of the most innovative people on the planet in **2004** when he launched **Facebook**.

⟨**NBA, 1946**⟩: The Capitols were founded in **1946** as a charter member of the Basketball Association of America, which became the **NBA** in 1949.

⟨**ISS, 1998**⟩: The first component of the **ISS** was launched in **1998**, and the space station has had ammonia leak problems before.

## A.2 The New York Times Annotated Corpus

**P22 father**

⟨**Kim Jong Il, Kim Il Sung**⟩: One sign that the North may be ready to consider a thaw is that its leader, **Kim Jong Il**, the son of the late President **Kim Il Sung**, formally took over the leadership of the ruling party this year.

⟨**Peyton Manning, Archie Manning**⟩: **Archie Manning** believes Parcells will pick his son, Tennessee quarterback **Peyton Manning**, as the top choice in the draft.

⟨**Tiger Woods, Earl Woods**⟩: With his father, **Earl**, and mother, Kultida, watching in the gallery, **Woods** coolly navigated the course as if he felt he would win all along.

⟨**Birendra of Nepal, Mahendra of Nepal**⟩: The experiment ended when **King Mahendra**, the father of the present monarch, **Birendra**, dismissed the Government in 1961.

⟨**Jean-Claude Duvalier, Francois Duvalie**⟩: April 21, 1971 **Francois Duvalier** dies and his 19-year-old son, **Jean-Claude Duvalier**.

⟨**Benjamin Cheever, John Cheever**⟩: The Plagiarist by **Benjamin Cheever**, the late **John Cheever**'s son, of Pleasantville (Atheneum).

⟨**Lee Hsien Loong, Lee Kuan Yew**⟩: Ms. Ho is married to Deputy Prime Minister **Lee Hsien Loong**, son of the founding prime minister and now senior minister, **Lee Kuan Yew**.

⟨**Martha McPhee, John McPhee**⟩: The other two fiction nominees are "Gorgeous Lies" (Harcourt), about a dying patriarch and his family, by **Martha McPhee**, a daughter of the writer **John McPhee**, and "Big If," (Norton), a Washington novel of intrigue and assassination by Mark Costello.

⟨**Indira Gandhi, Jawaharlal Nehru**⟩: But he spent much of his time maneuvering to keep his tottering coalition together, drawing disparaging comparisons from political commentators to the more charismatic leaders of the past, including **Jawaharlal Nehru** and his daughter, **Indira Gandhi**.

⟨**Karenna Gore Schiff, Al Gore**⟩: This weekend, the President cut short his European trip by a day and on Saturday flew back to Washington, where he attended the wedding reception of Mr. **Gore**'s daughter, **Karenna**.

**P25 mother**

⟨**Michael Kennedy, Ethel Kennedy**⟩: **Michael Kennedy** said his mother, **ETHEL KENNEDY**, had attended the small private ceremony that was performed by Vardis Vardinoyannis, the owner and captain of the ship, the Varmar VE.

⟨**Jesus, Michael Kennedy**⟩: **Mary**, the mother of **Jesus**, who is said to have saved Itri from a plague in 1527.

⟨**Prince William(Duke of Cambridge), Diana(Princess of Wales)**⟩: Others questioned the announcement that Charles and **Diana** would share custody of their children, Princes **William**, 10, and Harry, 8.

⟨**Prince William(Duke of Cambridge), Diana(Princess of Wales)**⟩: The lives of princes **William** and Harry, before and after the death of their mother, **Princess Diana**.

⟨**Rajiv Gandhi, Feroze Gandhi**⟩: Mrs. **Gandhi**'s son, **Rajiv**, who became prime minister after her death, was killed by a Tamil suicide bomber in 1991.

⟨**Michael Reagan, Jane Wyman**⟩: But the speech did not engender good feelings in **Michael Reagan**, a conservative talk-show host, who is the son of the late president and his first wife, **Jane Wyman**.

⟨**Hyatt Bass, Anne Bass**⟩: The only other project he has completed in the United States is the renovation of a town house in Greenwich Village for **Hyatt Bass**, a daughter of Sid and **Anne Bass**, and her husband, Josh Klausner.

⟨**Kiran Desai, Anita Desai**⟩: Hilma Wolitzer's daughter Meg is a novelist, as is **Anita Desai**'s daughter **Kiran**, whose second book just won the Booker Prize – an award that has so far eluded her mother.

⟨**Amy Carter, Rosalynn Carter**⟩: **Amy Carter** was listed in good condition today at a hospital here after she hurt her back on a visit here with her parents, former President Jimmy Carter and his wife, **Rosalynn**.

⟨**Snorri Thorfinnsson, Gudrid Thorbjarnardóttir**⟩: They stayed three years, and his wife, **Gudrid**, gave birth to a boy, **Snorri**, presumably the first European born in America.

**P26 spouse**

⟨**Hillary Clinton, Bill Clinton**⟩: In a January 1994 confidential memorandum to Mr. **Clinton**'s wife, **Hillary**, and Bruce Lindsey, a longtime Clinton friend in the White House counsel's office, Ms. Scott expressed concern about involving employees in the upkeep of the system.

⟨**Eli Wallach, Anne Jackson**⟩: Mr. **Wallach**, whose wife, **Anne Jackson**, is not Jewish, has a Christmas tree in his home as well as a menorah.

⟨**Jane Fonda, Ted Turner**⟩: Ms. **Fonda** was joined by her husband, **TED TURNER**, the head of the Turner Broadcasting System, which also owns Cable News Network.

⟨**Rostropovich, Galina Vishnevskaya**⟩: Mr. **Rostropovich**, accompanied by his wife, the opera singer **GALINA VISHNEVSKAYA**, received a tumultuous welcome at Sheremetevo Airport, where hundreds of friends, fans and journalists gathered to meet his flight from Tokyo, The Associated Press reported.

⟨**Yelena Bonner, Andrei Sakharov**⟩: He also was a vocal supporter of the physicist **Andrei Sakharov** and his wife, **Yelena Bonner**.

⟨**Benazir Bhutto, Asif Ali Zardari**⟩: Two versions, both denied by Ms. **Bhutto**, say she will either leave politics in return for immunity or send her husband, **Asif Ali Zardari**, into exile.

⟨**James Carville, Mary Matalin**⟩: Ms. **Matalin** was recently married to **James Carville**, her counterpart in the Clinton campaign.

⟨**Donald J. Trump, Marla Maples**⟩: But certainly the lesson to be taken from the news that the marriage of **Donald J. Trump** and **Marla Maples** is on the rocks.

⟨**Diana(Princess of Wales), Charles(Prince of Wales)**⟩: She then recalled that the last time she'd wakened early for a televised event was for **Princess Diana**'s marriage to **Prince Charles**.

⟨**Michael Jackson, Lisa Marie Presley**⟩: Not long after, the wedding of **Lisa Marie Presley** and **Michael Jackson** became a matter of law in the Dominican Republic.

**P3373 sibling**

⟨**Theodore Kaczynski, David Kaczynski**⟩: We can be indebted to **David Kaczynski** for helping the authorities apprehend his brother, **Theodore Kaczynski**, the suspect in the Unabom case.

⟨**Felix Mendelssohn, Fanny Mendelssohn**⟩: Its goal is to record all the works of **Felix Mendelssohn**, and **Fanny Mendelssohn Hensel**, his sister.

⟨**Felix Mendelssohn, Fanny Mendelssohn**⟩: Most of **Fanny Mendelssohn**'s compositions were published under the name of her brother, **Felix**, for the same reason.

⟨**George Gershwin, Ira Gershwin**⟩: Before **George Gershwin**'s death at the age of 38 in 1937, he and his brother **Ira** wrote more than 700 songs for stage and screen.

⟨**George W. Bush, Jeb Bush**⟩: Gov. **George W. Bush** of Texas, who easily won re-election, and his brother, **Jeb**, who was elected Governor of Florida.

⟨**Bob Weinstein, Harvey Weinstein**⟩: **Harvey Weinstein**, the co-chairman with his brother, **Bob**, of Miramax, the most prolific independent film company, which released "Pulp Fiction," acknowledged that although he was entirely independent of Disney, which bought the company three years ago, he had set some self-imposed creative limits.

⟨**Serena Williams, Venus Williams**⟩: The fourth-round match was won by the elder sibling, **Venus Williams**, 7-6 (5), 6-2 against her sister **Serena**.

⟨**John J. Gotti, Peter Gotti**⟩: The case against the Nassos stemmed from a larger case against **Peter Gotti**, a brother of the deceased mob leader **John J. Gotti**, and six other men who prosecutors said were members of the Gambino crime family.

⟨**Corin Redgrave, Vanessa Redgrave**⟩: Her parents, the late Michael Redgrave and Rachel Kempson, were luminaries on the English stage, and all three of their children – **Corin**, **Vanessa** and Lynn – followed in their footsteps.

⟨**Gianni Versace, Donatella Versace**⟩: **Gianni Versace** and his siblings, **Donatella** and Santo, could only look up, just as generations of Italian immigrants could only look west toward America.

**P40 child**

⟨**Isaac, Esau**⟩: Jacob, the second-born twin son of **Isaac**, cheats his brother, **Esau**, twice – first of his birthright and then of his dying father's blessing.

⟨**Elvis Presley, Lisa Marie Presley**⟩: **Lisa Marie Presley**, daughter of **Elvis Presley**, moves into home near Scientology retreat in Clearwater, Fla; has been longtime member of church (Chronicle).

⟨**Norodom Sihanouk, Norodom Ranariddh**⟩: The party was founded by Prince **Sihanouk** but is now led by one of his sons, Prince **Norodom Ranariddh**.

⟨**Arie Luyendyk, Arie Luyendyk, Jr.**⟩: **Luyendyk** said, referring to **Arie Jr.**, the oldest of his four children.

⟨**Bob Dylan, Jakob Dylan**⟩: **Bob Dylan** and his son, **Jakob Dylan** (leader of the Wallflowers), each received three nominations.

⟨**George Steinbrenner, Hal Steinbrenner**⟩: **Steinbrenner**'s sons, **Hal** and Hank, and his son-in-law, Steve Swindal, are the Yankees' general partners.

⟨**Andrew Wyeth, Jamie Wyeth**⟩: His grandson **Jamie**, whose father is the artist **Andrew Wyeth**, is known for his portraiture, images of everyday scenes and illustrations for children's books.

⟨**Ingrid Bergman, Isabella Rossellini**⟩: **Isabella Rossellini** broke into the business largely because her parents were **Ingrid Bergman** and Roberto Rossellini, although it was her own beauty that ultimately landed her in many cosmetics advertisements.

⟨**Alfred Lerner, Randolph Lerner**⟩: Mrs. Lerner is the widow of MBNA's former chairman, **Alfred Lerner**, and the mother of **Randolph Lerner**, who succeeded his father as chairman in late 2002.

⟨**George VI, Elizabeth II**⟩: **Elizabeth II** became the monarch only because her father, King **George VI**, died without sons.

**P112 foundedBy**

⟨**Next Inc., Steven P. Jobs**⟩: **Next Inc.**, the company started by **Steven P. Jobs**, who co-founded Apple Computer, incorporates the technique in software for its computer to make it easier for software companies to write programs for the machine.

⟨**Apple Inc., Steven Jobs**⟩: And cleaning its own house, **Apple** changed its board, bringing in **Steven P. Jobs**, its co-founder, below, to a more official role and sweeping out A. C. Markkula Jr., another co-founder and power broker.

⟨**Creative Artists Agency, Michael Ovitz**⟩: **Michael Ovitz**, the co-founder of **Creative Artists Agency**.

⟨**Creative Artists Agency, Michael Ovitz**⟩: Mr. **Ovitz**, who as co-founder and talent agent of **Creative Artists Agency**, was dubbed "the most powerful man in Hollywood" by the media, is expected to lead off the defense, possibly late Monday.

⟨**Miramax, Harvey Weinstein**⟩: "You have to taste the sweat," says **Harvey Weinstein**, **Miramax**'s co-founder and co-chairman

⟨**DreamWorks, Jeffrey Katzenberg**⟩: What has yet to be worked out is whether Mr. Spielberg, **Jeffrey Katzenberg** and David Geffen, the **DreamWorks** co-founders, and their largest investor, Paul G. Allen, would be paid right away or be asked to share instead in future revenues.

⟨**DreamWorks, David Geffen**⟩: Behind Talks In Hollywood, Longtime PalsRon Meyer, president of Universal Studios, is longtime friend of **David Geffen**, co-founder of **DreamWorks SKG**, which NBC Universal is in talks to acquire; Meyer's interest in Dreamworks goes back more than a year and became more intense after recent telephone conversation between two men.

⟨**Microsoft, Bill Gates**⟩: **William H. Gates**, the company's chairman and co-founder, hailed the new program as "the best operating system **Microsoft** has ever built."

⟨**Microsoft, Bill Gates**⟩: His involvement may have stemmed from a desire to trump one of his present or former business rivals, the co-founders of **Microsoft**, **Bill Gates** and Paul Allen.

⟨**eBay, Pierre Omidyar**⟩: Virtual world proponents – including a roster of Linden Labs investors that includes Jeffrey P. Bezos, the founder of Amazon.com; Mitchell D. Kapor, the software pioneer; and **Pierre Omidyar**, the **eBay** co-founder – say that the entire Internet is moving toward being a three-dimensional experience that will become more realistic as computing technology advances.

**P169 CEO**

⟨**Apple Inc., Gil Amelio**⟩: Apple Getting Reactions On the WebArticle discusses Web sites dedicated to news about **Apple Computer Inc**; reaction to resignation last week of chief executive **Gilbert Amelio** discussed.

⟨**Cendant, Henry Silverman**⟩: Mr. **Silverman**'s son, Henry R. **Silverman**, Chairman, President, and CEO of **Cendant Corporation**, also serves on our board.

⟨**IBM, Thomas J. Watson Jr.**⟩: Shortly after **Thomas J. Watson Jr.** became President and CEO of **IBM** in 1956, he reorganized the company in anticipation of its probable explosive growth following the introduction of computers.

⟨**Nissan, Carlos Ghosn**⟩: In the letter, filed with the U.S. Securities and Exchange Commission, Tracinda said it has been in talks with **Nissan** CEO **Carlos Ghosn**.

⟨**The Walt Disney Company, Michael Eisner**⟩: "If you look at it, it's back to having a flair," said **Michael D. Eisner**, the chief executive of **the Walt Disney Company**.

⟨**The Walt Disney Company, Michael Eisner**⟩: Mr. **Eisner**, **Disney**'s chairman and chief executive, is well protected.

⟨**Apple Inc., John Sculley**⟩: "Based on our company's favorable prospects, we believe purchase of Apple shares enhances the value to our shareholders," said **John Sculley**, **Apple**'s chairman and chief executive.

⟨**Apple Inc., John Sculley**⟩: Mr. Markkula worked at **Apple** for many years until he was replaced as chief executive by **John Sculley** in 1983.

⟨**Lehman, Richard S. Fuld**⟩: **Richard S. Fuld**, **Lehman**'s chairman and chief executive, said in a statement.

⟨**CBS Corporation, Les Moonves**⟩: For **Leslie Moonves**, the chief executive of the **CBS Corporation**, it was a week to savor.

**P69 educatedAt**

⟨**DePauw University, Quayle**⟩: The dispute revolves around the events of 1969, just after Mr. **Quayle**'s graduation from **DePauw University**.

⟨**Yale University, Oliver Stone**⟩: **Oliver Stone** dropped out of **Yale** and ended up in Vietnam.

⟨**Harvard University, Ellen Johnson-Sirleaf**⟩: His closest rival, **Ellen Johnson-Sirleaf**, graduated from **Harvard** and later worked as a Citibank executive.

⟨**Cornell University, Michael Schwerner**⟩: **Michael Schwerner**, a **Cornell** graduate, had gone to Mississippi with his wife, Rita, to work for the Congress of Racial Equality.

⟨**Kansas State College, Frank Carlson**⟩: **Frank Carlson** attended Concordia Normal and Business School and **Kansas State College**, majoring in agriculture.

⟨**Royal Ballet School, Christopher Wheeldon**⟩: Mr. **Wheeldon**, 23, a former student of **the Royal Ballet School** in London, has choreographed works for the Royal Ballet and the School of American Ballet.

⟨ **Stanford University, Tiger Woods**⟩: That was the high point for **Woods**, the 20-year-old amateur from Cypress, Calif., who will be entering his junior year at **Stanford** in the fall.

⟨**Georgetown University, Bill Clinton**⟩: There as well was President **Clinton**, whose senior prom at **Georgetown University** was canceled because of the assassination, and who was drawn into public life in large part by the Kennedys' example.

⟨**Ohio State University, Maurice Clarett**⟩: Last year at this time, as **Maurice Clarett** began his sophomore year at **Ohio State**, television cameras followed him around campus.

⟨**Rhode Island School of Design, Chris Frantz**⟩: He eventually studied art at **the Rhode Island School of Design**, where he met the bassist Tina Weymouth and the drummer **Chris Frantz**, with whom, after moving to New York, he formed Talking Heads in 1975.

**P26P580 yearOfMarriage**

⟨**Bruce Nauman, 1989**⟩: Mr. **Nauman**, whom she married in **1989**, may be the most influential artist today; he raises horses and cattle when he isn't making art.

⟨**Ivana Winklmayr, 1977**⟩: She also reintroduces us to **Ivana Winklmayr**, the beauty from Czechoslovakia whom Trump married in **1977**.

⟨**Bill Clinton, 1975**⟩: On Oct. 11, **1975**, **Bill Clinton** and Hillary Rodham married

⟨**Phyllis Newman, 1960**⟩: Mr. Green had two unsuccessful marriages before marrying the actress **Phyllis Newman** in **1960**.

⟨**Charles(Prince of Wales), 1981**⟩: Diana, after all, is of royal blood, and was demure when, at age 19, she married **Prince Charles** in **1981**.

⟨**Diana(Princess of Wales), 1981**⟩: Touristy mugs from Prince Charles's **1981** wedding to **Diana** and Princess Anne's 1973 wedding to Capt.

⟨**Hillary Clinton, 1975**⟩: On Oct. 11, **1975**, Bill Clinton and **Hillary Rodham** married.

⟨**Anaïs Nin, 1923**⟩: For all that time, **Nin** was married to her first husband, Hugh Guiler, whom she wed in **1923**.

⟨**Hussein of Jordan, 1961**⟩: She also supported **King Hussein**'s decision to marry a Briton, Antoinette Gardiner, as his second wife, in **1961**.

⟨**Vanessa Redgrave, 1962**⟩: Both are directed by Tony Richardson, who would marry Michael Redgrave's daughter **Vanessa** in **1962**.

### P569 yearOfBirth

⟨**Henry Roth, 1906**⟩: **Henry Roth** was born Feb. 8, **1906**, in Tysmenica, Galicia, in what came to be called Ukraine.

⟨**Franz Schubert, 1797**⟩: The "Stabat Mater" of **Franz Schubert** commemorates the bicentennial of his birth in **1797**.

⟨**Pat Hingle, 1924**⟩: **Pat Hingle** was born in Miami in **1924** as Martin Patterson Hingle.

⟨**Karl Blossfeldt, 1865**⟩: **Karl Blossfeldt** (**1865**-1932).

⟨**Maxwell George Lorimer, 1908**⟩: Mr. Wall, whose name was originally **Maxwell George Lorimer**, was born in south London on March 12, **1908**.

⟨**Hildegard of Bingen, 1098, **⟩: There will be tributes to George Gershwin (born 1898), Hanns Eisler (1898) and **Hildegard of Bingen** (**1098**), and 30 world, European, British and London premieres.

⟨**Emily Dickinson, 1830**⟩: This was the home where **Emily Dickinson** was born in **1830**, lived most of her life and died 56 years later.

⟨**Alfred Zucker, 1852**⟩: Built in 1895, the Bolkenhayn was designed and owned by **Alfred Zucker**, who was born in **1852** in the Silesia region of Prussia.

⟨**Peter Gzowski, 1934**⟩: **Peter Gzowski** was born in Toronto on July 13, **1934**, the only son of Margaret Brown and Harold Gzowski.

⟨**Walt Whitman, 1819**⟩: Professor Allen was the son of a carpenter, as was **Whitman**, who lived from **1819** to 1892 and is widely considered the greatest American poet.

**P570 yearOfDeath**

⟨**Jacqueline Kennedy Onassis, 1994**⟩: The 14-room Fifth Avenue co-op owned by Jacqueline Kennedy Onassis was put on the market for $9 million after Mrs. **Onassis**'s death in **1994**.

⟨**Jacqueline Kennedy Onassis, 1994**⟩: In **1994**, the reservoir was officially named for **Jacqueline Kennedy Onassis**, who had died that year.

⟨**Blackbeard, 1718**⟩: Somewhere within my field of vision was the very spot where **Blackbeard** died on Nov. 21, **1718**, after boarding Maynard's vessel.

⟨**Salvador Allende, 1973**⟩: After Dr. **Allende**'s death on Sept. 11, **1973**, in a palace under bombardment by the armed forces - his personal surgeon, who was present, said he had committed suicide - his body was buried, without identification, in the mausoleum of some of his in-laws in a cemetery in Vina del Mar.

⟨**Jonathan Larson, 1996**⟩: Mr. **Larson**, 35, died of a heart aneurysm in January **1996**.

⟨**Dietrich Bonhoeffer, 1945**⟩: **Bonhoeffer** was hanged, as were other members of the Resistance who had tried to assassinate Hitler, in April **1945**.

⟨**Titian Ramsay Peale, 1885**⟩: As an artist, **Titian Ramsay Peale** (1799-**1885**) made all of nature his subject.

⟨**Thomas Hale Boggs, Sr. , 1972**⟩: After Mr. **Boggs** died in a plane crash in Alaska in **1972**, Mr. O'Neill was elected majority leader.

⟨**Guy Lombardo, 1977**⟩: Mr. Gardner said in a 1994 video tribute to **Guy Lombardo**, who died in **1977**.

⟨**Edward Thatch, 1718**⟩: On Nov. 22, **1718**, **Edward Thatch**, better known as Blackbeard, was killed in a bloody shipboard battle.

**P571 yearFounded**

⟨**Friendly's, 1935**⟩: Founded in **1935**, **Friendly's** has long been an industry leader, offering innovations like, in 1951, half-gallons of ice cream that consumers could take home.

⟨**Wieden & Kennedy, 1982**⟩: **Wieden & Kennedy** has been the lead worldwide creative agency for Nike brand advertising from **1982** to 1984 and from 1985 on.

⟨**Hamas, 1987**⟩: **Hamas** was founded on Dec. 14, **1987**, just days after the outbreak of the Palestinian uprising.

⟨**The Farnsworth House, 1951**⟩: **The Farnsworth House** began in **1951** as a private retreat, but it deserves now to become a public pavilion.

⟨**PRI, 1929**⟩: And the party was the Institutional Revolutionary Party, known as the **PRI**, which has governed since its founding in **1929**.

⟨**Blue Origin, 2000**⟩: In **2000**, he registered a company, **Blue Origin**, and hired rocket scientists.

⟨**Bethlehem Steel, 1857**⟩: **Bethlehem Steel** was founded in **1857** and was once among the largest and most profitable companies in the country.

⟨**XACBank, 1998**⟩: **XACBank** in Mongolia was started in **1998** as XAC by two development agencies at the United Nations, not by Mercy Corps.

⟨**Cigna, 1982**⟩: **Cigna** was formed in **1982** with the merger of the INA Corporation and the Connecticut General Corporation.

⟨**Fox News, 1996**⟩: He created **Fox News** in **1996** when CNN, the hegemon of cable news, seemed to reign unchallenged.

# Appendix B

# Recalls @ aN for All Relations

| Features | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| Orig | 6.50 | 13.00 | 27.91 | 56.81 |
| -Tfidf | **7.50** | **15.00** | 30.16 | **57.33** |
| -Syn | 6.45 | 12.90 | 28.59 | 56.01 |
| -ULex | 7.40 | 14.80 | **33.03** | 53.33 |
| -BLex | 6.60 | 13.20 | 28.87 | 54.15 |
| +WVec | 5.90 | 11.80 | 23.20 | 49.13 |

Table B.1: Recall @ aN for the *P22 father* relation.

| Features | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| Orig | **3.65** | **7.30** | **14.60** | **27.40** |
| -Tfidf | 3.50 | 7.00 | 14.00 | 26.35 |
| -Syn | 2.70 | 5.40 | 10.80 | 21.84 |
| -ULex | 3.60 | 7.20 | 14.40 | 27.07 |
| -BLex | **3.65** | **7.30** | **14.60** | 26.85 |
| +WVec | 3.45 | 6.90 | 13.80 | 26.50 |

Table B.2: Recall @ aN for the *P25 mother* relation.

| Features | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| Orig | 19.05 | 34.84 | 54.91 | 64.31 |
| -Tfidf | 18.75 | 34.41 | 55.29 | **64.37** |
| -Syn | 19.10 | 35.40 | 55.40 | 64.11 |
| -ULex | 19.15 | 34.96 | 54.24 | 63.77 |
| -BLex | **19.35** | **35.76** | **56.79** | **64.37** |
| +WVec | 18.65 | 33.53 | 51.91 | 63.96 |

Table B.3: Recall @ aN for the *P26 spouse* relation.

| Features | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| Orig | 11.25 | 22.50 | **38.86** | **53.14** |
| -Tfidf | 11.25 | 22.50 | 38.80 | 52.89 |
| -Syn | 10.20 | 20.40 | 35.59 | 51.02 |
| -ULex | 11.70 | 23.40 | 38.59 | 48.94 |
| -BLex | **11.80** | **23.60** | 38.78 | 49.46 |
| +WVec | 10.10 | 20.20 | 35.65 | 51.25 |

Table B.4: Recall @ aN for the *P3373 sibling* relation.

| Features | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| Orig | 10.65 | 21.42 | 42.01 | **61.85** |
| -Tfidf | **11.05** | **22.25** | **42.84** | 61.37 |
| -Syn | 9.15 | 18.54 | 38.48 | 60.14 |
| -ULex | 9.30 | 18.67 | 36.08 | 55.55 |
| -BLex | 10.20 | 20.60 | 40.30 | 58.66 |
| +WVec | 10.55 | 21.31 | 41.41 | 59.84 |

Table B.5: Recall @ aN for the *P40 child* relation.

| Features | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| Orig | 28.30 | 43.85 | 55.70 | 63.95 |
| -Tfidf | 27.60 | 43.07 | 55.38 | 64.10 |
| -Syn | 26.85 | 41.90 | 55.94 | 63.77 |
| -ULex | 27.90 | 43.53 | 54.04 | 63.91 |
| -BLex | **28.85** | **45.28** | **56.19** | **66.92** |
| +WVec | 27.75 | 42.93 | 53.86 | 64.20 |

Table B.6: Recall @ aN for the *P112 foundedBy* relation.

| Features | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| Orig | **11.80** | **23.60** | **47.20** | 68.01 |
| -Tfidf | 11.70 | 23.40 | 46.80 | **68.88** |
| -Syn | 11.60 | 23.20 | 46.40 | 66.27 |
| -ULex | 9.00 | 18.00 | 36.00 | 56.30 |
| -BLex | 11.75 | 23.50 | 47.00 | 67.47 |
| +WVec | 11.00 | 22.00 | 44.00 | 63.55 |

Table B.7: Recall @ aN for the *P169 CEO* relation.

| Features | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| Orig | 21.66 | **38.74** | **61.69** | 75.57 |
| -Tfidf | 21.82 | 38.69 | 61.03 | 75.59 |
| -Syn | 21.43 | 37.25 | 59.31 | 73.90 |
| -ULex | 19.90 | 36.25 | 55.71 | 74.17 |
| -BLex | **21.93** | 38.04 | 60.28 | **75.94** |
| +WVec | 19.49 | 34.99 | 54.88 | 72.18 |

Table B.8: Recall @ aN for the *P69 educatedAt* relation.

| Features | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| Orig | **19.50** | **34.22** | 35.82 | **39.42** |
| -Tfidf | **19.50** | 34.20 | 35.83 | 39.36 |
| -Syn | **19.50** | 34.18 | 35.71 | 39.01 |
| -ULex | **19.50** | 34.18 | **35.92** | 39.01 |
| -BLex | **19.50** | **34.22** | 35.85 | 38.80 |
| +WVec | **19.50** | 34.14 | 35.67 | 39.01 |

Table B.9: Recall @ aN for the *P26P580 yearOfMarriage* relation.

| Features | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| Orig | **61.56** | **64.55** | **65.83** | 67.49 |
| -Tfidf | 61.17 | **64.55** | 65.33 | 67.48 |
| -Syn | 60.47 | **64.55** | 65.80 | 67.52 |
| -ULex | 58.05 | 64.42 | 65.23 | **67.55** |
| -BLex | 61.28 | 64.51 | 65.81 | 67.47 |
| +WVec | 61.22 | **64.55** | 65.80 | 67.49 |

Table B.10: Recall @ aN for the *P569 yearOfBirth* relation.

| Features | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| Orig | 46.55 | 58.71 | 61.64 | **64.93** |
| -Tfidf | **47.10** | **58.78** | 61.74 | **64.93** |
| -Syn | 46.57 | 58.68 | 61.64 | 64.90 |
| -ULex | 43.06 | 58.23 | 61.51 | 64.89 |
| -BLex | 46.46 | 58.71 | **61.84** | 64.91 |
| +WVec | 45.56 | 58.57 | 61.71 | 64.84 |

Table B.11: Recall @ aN for the *P570 yearOfDeath* relation.

| Features | Recall @ | | | |
|---|---|---|---|---|
| | 3N | 6N | 12N | 24N |
| Orig | **33.38** | **41.18** | 46.82 | 60.55 |
| -Tfidf | 32.85 | 41.04 | 46.73 | 60.44 |
| -Syn | 30.77 | 40.42 | 46.44 | 60.54 |
| -ULex | 28.52 | 39.90 | 46.54 | 58.20 |
| -BLex | 31.06 | 40.79 | **47.02** | 60.36 |
| +WVec | 32.11 | 40.45 | 46.53 | **60.57** |

Table B.12: Recall @ aN for the *P571 yearFounded* relation.