Answer Extraction Module for Question Answering Gaming Application

a thesis presented

by

Desmond Darma Putra

October, 2013

UNIVERSITÄT DES SAARLANDES

THE UNIVERSITY OF MELBOURNE

Erasmus Mundus European Master in Language & Communication Technologies (LCT)

Saarland University, University of Melbourne

Thesis advisors
Prof. Dietrich Klakow
Dr. Volha Petukhova
A/Prof Tim Baldwin

Author
Desmond Darma Putra

Abstract

Question answering is a Natural Language Processing field that focuses on providing answer to user's question, posed in natural language. In the project that we work, we aim to build a Question Answering Dialogue System that creates an interactive game between the user and the system. In the game scenario, the system will hold a famous person's biographical facts and the user has to guess the name of this famous person. To achieve this goal, the user may ask up to ten question. However, the system prevents the user from asking direct questions about the person's name or alias.

This system consists of four modules named Interpretation Module, Dialogue Manager, Answer Extraction Module, and Utterance Generation Module. The focus of this thesis is to build Answer Extraction module (AEM) so it can automatically extract these biographical facts from raw texts. The texts are mainly gathered from Wikipedia. In order for the system to understand the texts, 53 semantic relations are defined to extract most important facts from them. Texts are annotated with defined semantic relations adding IOB (Inside, Outside, Beginning) prefixes to capture expected answer's boundaries.

Our Answer Extraction Module is comprised of three components: sequence classifiers, pattern matching tool and post processing pipeline. For sequence classifiers, we employ two well-known machine-learning algorithms, which are SVMs and CRFs. Features such as word token, lemma, Part-of-Speech tag, Named Entity tag, chunk information, capitalization and keyword are used to train the classifiers. The learning model which is the outcome of this training is later used to predict the relations' label. For pattern matching, handcrafted regular expressions are applied with the same task as the classification one. The output from the classifiers and pattern matching are then validated through the post processing pipeline.

The performance of our AEM is investigated in two experiment sets. In the first experiment, we use five-fold cross validation and apply three systems i.e. the Baseline, System 1 and System 2. Our results show that System 2 outperforms the baseline and System 1. The reasons are threefold. First, developing the classifiers to predict each relation separately gives more robust performance than the one that predict all relations together. Second, for several relations, pattern matching works better than the classifiers. Third, post processing pipeline is proven to filter and remove the incorrect output. In the second experiment, different sizes of training set (20%, 40%, 60 and 80%) are used to see how the classifiers work. The results demonstrate that the more training data is used, the better the classifiers' performance.

Declaration

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Saarbrücken, 10 October 2013

Desmond Darma Putra

Acknowledgement

First of all, I would like to thank The God almighty who answered my prayer to study in this LCT program.

I would like to thank my supervisor Dr. Volha Petukhova, for her guidance, inspiration, support and discussion throughout the thesis work. Without her, maybe I would not finish this thesis on time. In addition, I would like to thank Prof. Dietrich Klakow for his idea and spirit throughout the whole process. I also would like to thank to A/Prof Tim Baldwin for his guidance throughout my first year in Melbourne University.

Additionally, I would like to thank Martin Gropp for his help in the technical stuff especially related to Linux. I would like also to thank to Benjamin Roth for sharing his idea about TAC KBP Slot filling task. For the rest of the LSV members, I would also like to thank for their hospitality and support that are given until now.

I also want to thank my family, my father, mother and brother who always pray and support me from Indonesian. In addition, I would also thank to Jessica who has been so supportive during my master study in Australia and Germany.

Finally, I would like to thank EM LCT for giving me the opportunity to purse the study in this field and to study in two excellent universities. Thanks to Ivana and Bobbye for their support throughout these two years. I would like to thank Richard as a LCT student representative for his help in the LCT in general and also his help with the grammar checking. Last, I want to say thank to Victor, Lili, Marina, Adam, Matej, Igor, and the rest of LST students for the great company and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Question answering (QA) is one of the Natural Language Processing (NLP) fields, which is concerned with providing an answer automatically as a response to a question that is posed by humans using natural language. This field has attracted a lot of researchers attention since 1970, with the end goal of developing a system that can retrieve an answer for any given question.

In the early start of QA, the system could only dealt with factoid questions. '*When was Obama born?*' or '*Who did invent the telephone?*' are examples of such factoid questions. In order to find an answer to this sort of question, the system usually used Named Entity Recognition (NER) tagger to detect the entities (e.g. person, location, time, organization) in the documents.

With increasing computer power and also with the existence of internet, this QA system able to handle more complex questions. For examples, '*what is UFO?*' or '*List all the books that are written by J.K. Rowling?*' are considered as complex questions because the answers are not available in a form of simple Named Entities (NE) chunk. To find the correct answers, the system has to look into the documents and maybe summarize the passages before providing the final answer.

In addition to different question types, there are two separate domains in QA: closed domain and open domain. In the closed domain QA, the question only focuses on a certain topic and if there is a question outside that topic, it will be ignored by the system. On the other hand, open domain QA handles any kind of questions that a person may ask about. Finding an answer in open domain, in general, takes more time because the system needs to process a large amount of data.

Besides the domain and complexity of the question types, this field also involves other NLP fields such as information retrieval and information extraction. Information retrieval focuses on providing a list of documents that are relevant to some query. These documents

are subsequently passed to the information extraction module to extract the passages or text segments that satisfy users' information needs.

There are at least two ways to evaluate the quality of QA system. First, Mean Reciprocal Rate (MRR) is used to calculate the average of reciprocal ranks. This reciprocal rank is a multiplicative inverse of the rank of the first correct answer. For example, if the system provides five answers but the first two answers are incorrect and the first correct answer is in the third position, then the reciprocal rank for that question is 1/3. Furthermore, the score for each question is averaged by the number of questions asked. Second, we can evaluate the quality of a QA system by employing F-score to measure the accuracy of an answer. This F-score is further explained in Section 6.3.

1.1 QA paradigms

Currently, there are three common paradigms used to develop a QA system. The first paradigm is an Information-Retrieval (IR) based approach, which consists of three components: question processing, passage retrieval, and answer extraction. The first component task is to read the user's question and then detect the question type and the focus word of the question, and then to decide the expected answer type. This information is used later as a query to the second component. This component will utilize a search engine to retrieve a list of documents that are relevant to the query. In addition, this component also narrows the documents into a set of passages/paragraphs and then performs a re-ranking calculation to sort the confidence score, starting from the highest ranked item. The last component will process these passages, extract the candidate answers, and do another re-ranking mechanism of the candidate answers. Search engine such as Google[1], used this in their early development.

The second paradigm is knowledge-based approach that uses a semantic representation of the query and then maps it to query structured data such as Geospatial database, wikipedia infoboxes[2], etc. For example, if there is a question 'who is the daughter of Barack Obama?', the system would transform this question into (daughter-of 'Barack Obama' ?x) and find the answer into the database. The possible answer is formatted in the form of relation such as daughter-of('Barack Obama', 'Natasha Obama') and daughter-of('Barack Obama', 'Malia Ann Obama'). Some application examples that uses this paradigm are Apple Siri and Wolfram Alpha[3], among others.

The last paradigm is hybrid approach, which combines the previous two mentioned methods. This starts with building semantic representation of the query and then finds the

[1] www.google.com
[2] http://en.wikipedia.org/wiki/Help:Infobox
[3] http://www.wolframalpha.com/

answer candidates using the search engine, operating on additional ontologies and semi-structured data. Finally, the system scores each candidate answer based on knowledge resources. Some examples of such systems are IBM Watson, and Google.

1.2 Focus and Scope of this Thesis

For our gaming scenario, we aim to build an end-to-end Question Answering Dialogue System (QADS) that creates an interactive game between the user and the system. The system will hold biographical facts of a famous person and the user has to guess the famous person's name. This is considered as a closed domain QA since it concentrates only on biographical facts. To make the game more interactive, the user is allowed to ask up to ten questions to the system in order to find out the biographical description of this famous person. One constraint is that the system will prevent from answering user's question related to the name or alias of this famous person.

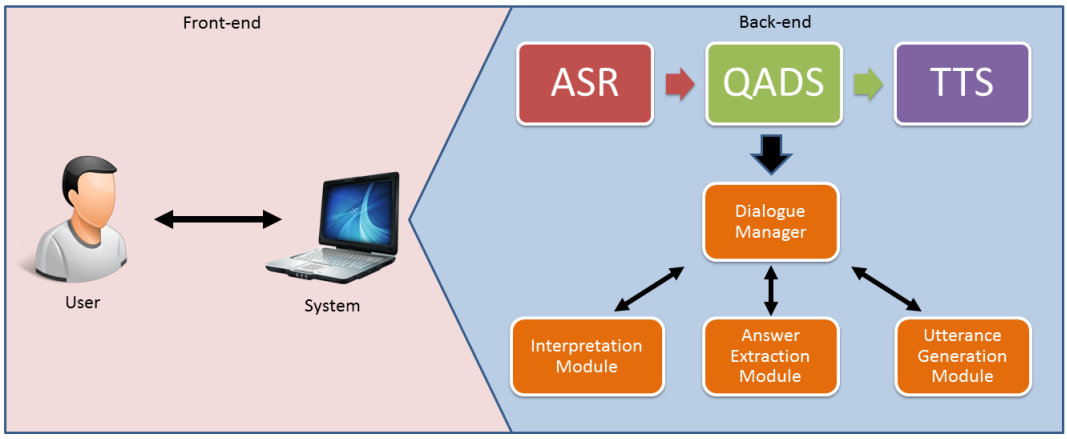

Figure 1.1: QADS architecture

Figure 1.1 shows our QADS architecture. The whole system would consist of three modules: Automatic Speech Recognition (ASR), our QADS, and Text-To-Speech (TTS) system. First, the words uttered by the user are received by the ASR, which provides hypotheses on the recognized tokens. Second, the ASR output is processed by our QADS to provide a response in the form of dialogue utterance. This output is used by TTS to generate the final spoken response. Nevertheless, both ASR and TTS system are not discussed here, because they are out of the scope of this thesis.

Our QADS adopts a statistical approach and consists of four major components: Dialogue Manager (DM), Interpretation Module (IM), Answer Extraction Module (AEM) and Utterance Generation Module (UGM). The DM takes care of the overall communication between the user and the system. It gets as input from the interpretation module a

dialogue act representation. Mostly it is about a question which is uttered by the human player. Questions are classified according to their communicative function (e.g. Propositional, Check, Set and Choice Questions) and semantic content. Semantic content is determined based on Expected Answer Type (EAT), e.g. LOCATION, and the focus word, e.g. *study*. To extract the requested information, 53 semantic relations were defined that cover most important facts in human life, e.g. birth, marriage, career, etc. The extracted information is mapped with the EAT and focus word, and the most relevant answer and the strategy how to continue the dialogue are computed. DM then passes the system response for generation, where the DM input is transformed into a dialogue utterance.

Our main focus here is to build an AEM so it can extract all biographical facts about the famous person automatically from raw texts. A challenge that we encounter immediately is that there is only a small amount of text is available. In addition, we do not have any guidelines on what information or facts should be extracted from the text. Moreover, the method used to extract these facts automatically needs to be explored.

To accommodate the above challenges, we define three main objectives that we want to achieve in this thesis:

1. Collect more data about famous people to create the dataset.
2. Define the semantic relations and annotate the dataset.
3. Create an automatic system that can extract these semantic relations automatically with a good accuracy.

1.3 Thesis Overview

The rest of the chapters is structured as follow:

Chapter 2 provides a background overview of some the techniques and approaches for QA system design. We summarize the evolution of QA tasks from two competitions such as TREC and TAC KBP slot filling. In addition, we explain two QA application games that are popular currently in order to understand the behavior of QA system.

Chapter 3 describes the concept of sequence classification and pattern matching for our purpose of creating an AEM. Furthermore, additional tools that support the classification task such as POS tagger, NER tagger, chunker are discussed in detail.

Chapter 4 explains how the semantic relations are defined to encode the biographical facts. Some of these relations are influenced by TAC KBP Slot filling task, while the rest

are defined for our task. Last, IOB-encoding concept is used to label the data with these relations.

Chapter 5 summarizes the data and our AEM architecture. Each component in the architecture such as the classifiers and pattern matching tool are presented. Moreover, we describe post processing pipeline to finalize the classifiers and pattern matching output.

Chapter 6 discusses two experiment sets and the evaluation metrics. The first experiment emphasizes on finding the best result, while the second experiment focuses on the learnability of the classifiers.

Chapter 7 presents and analyzes the experiments results. In addition, the system performance and the issues that the system has are also explained with some additional examples.

Chapter 8 summarizes the work performed and provides some suggestions for further research.

Chapter 2

Related Work

QA tasks have gained a lot of attention in the last several decades. One of the earliest work by Simmons et al. [1964] describes the use of dependency information and inverse frequency of the words to find an answer for a given question. Moving forward three decades later, a breakthrough in QA is done by the Text REtrieval Conference (TREC) and Text Analysis Conference (TAC). Detail explanation about these two events is provided in the next two sections. In addition, we will see two QA applications that are popular right now and see how they influence our work.

2.1 TREC

The Text REtrieval Conference (TREC) is a series of workshops co-sponsored by the National Institute of Standards and Technology (NIST) and U.S. Department of Defense. The purpose of this conference is to support Information Retrieval community by providing the infrastructure for large-scale evaluation of text retrieval. Each year, TREC has a number of tracks to deal with various information retrieval research areas such as QA track, microblog track, legal track, etc.

The QA track ran from 1998 and until 2007. From year to year, it has shown more sophisticated QA test sets from simple factoid questions to list questions, definition questions, and many more. In the first track (known as TREC 8), the system focused on finding the actual answer (text snippet) to a question rather than list of documents returned by the IR systems. Each participant was provided with a list of questions and text collections to find the answer. All the questions were factoid questions and the text collections are guaranteed to have the answers to these questions. The final answer (up to 5 pairs) contained the *id* of the document which it was retrieved from and the string (either 50 bytes

or 250 bytes format). To evaluate the performance, they applied Mean Reciprocal Rank (MRR).

Moving forward to 2001 track, the answer was distributed in more than one document. Therefore, each participant had to generate the answer from several documents in order create a complete answer. In addition, the answer was not guaranteed to be available in the text collections, thus the participant had to provide '*NIL*' if the answer was not found Burger et al. [2001]. For example:

Example 1. *Q1: what countries from the South America did the Pope visit and when?*[1]
Answer:
Argentina 1987 (document 1)
Columbia 1986 (document 3)
Brazil 1982, 1991 (document 4)

Q2: when did President Kennedy visit Cuba?
Answer: NIL

In 2003, list questions were introduced to the track with more complex form because the answer would not be a text snippet but it is more likely to be well-formed generated responses which have to resolve the ambiguity problem. For example:

Example 2. *Question: 'where is Taj Mahal?'.*
Answer:
If you are interested in the Indian Landmark, it is in Agra, India. There are several restaurants named Taj Mahal, A full list is rendered in the following places:
The Taj Mahal Indian Cuisine in Mountain View, CA.
The Taj Mahal Restaurant in Dallas, RX
Taj Mahal in Springfield, VA

Moreover, more complex questions were introduced. For example:

Example 3. *Question: 'How did Socrates die?'*
Answer: He drunk poisoned wine. Anyone drinking or eating something that is poisoned is likely to die.

In 2004 track, even more complex question were asked, that require answers to be summarized from different relevant document. '*how safe are commercial flights?*' and '*what other companies are operated with Government aid?*' are some examples of the questions.

A time frame was included in 2006 track in order to get the most up-to-date answer in the document collection. Questions were grouped into different sets and each set contained

[1]Example 1-4 are taken from Burger et al. [2001].

Answer Types				
Date	Location	NNP	Person	Time
Definition	Manner	Number	Price	Title
Distance	Money	Organization	Reason	Undefined

Table 2.1: Answer Types

around 8-9 questions. Furthermore, each question had already been tagged as factoid, list or other question type. Participants were allowed to use information from extracted answers from the previous questions in this set to help answering next questions in the same set. However, the participant was prohibited from looking into the later question to help answering an earlier question. An example of one question set with the focus of Barry Manilow can be seen below.

Example 4. *Barry Manilow:*

1. *FACTOID What year was he born?*
2. *FACTOID How many times has he married?*
3. *FACTOID What music school did he attend?*
4. *List List the songs he recorded?*

One of the work that is usually referred when discussing TREC comes from Moldovan et al. [2000]. They developed an end-to-end QA system for open domain which achieved the best result at the TREC-8 competition. Their system contains three components named question processing module, paragraph indexing module and answer processing module.

In question processing module, there are two important parts such as query formulation and answer type detection. In the first part, a list of words is extracted from a question that is used as a search query. The latter part employs the classifier to detect the expected answer type (e.g. person, location, etc.) of the question. The benefit of knowing the answer type is that the amount of time that is spent to check the output from the retrieval process can be reduced. For example, a question like 'who did invent cronuts?' is classified as person answer type. However, the answer type can be more sophisticated than just person, location or organization. In their work, they defined 15 classes for the answer type[2] (See Table 2.1).

[2]Later on, Li and Roth [2002] defined the taxonomy to handle numerous of answer types. They defined six coarse answer types: abbreviation, entity, description, human, location and numeric. Each type from these six is split into finer types. In total, there are 50 finer answer types.

The query that comes out from the query formulation is used by the IR engine to retrieve a set of web pages or documents that are relevant to it. The IR engine ranks these results based on the relevance. However, the final answer does not have to come from the first ranked document or web page. It is possible that the final answer is selected from the second, or third, even maybe tenth rank. Further processing is being done by passage retrieval to select only some passages from each document or web page that is relevant. This selection process can be done, for example by NER tagger, to find the passages that consist the answer type. After the passages are selected, they are ranked by some machine learning techniques using features such as number of NE, number of query words in the passage, etc.

After reranking the passages, the next process is to extract the answer. It can be done by applying pattern matching mechanism that is associated with the expected answer type and then finding this pattern in the passages. The matched output is later being checked using NER tagger to confirm the type. In addition, if the answer type cannot be detected using pattern matching and NER tagger, another possible attempt is to apply some machine learning tools to predict the final output given the answer type. These tools later select the best answer based on the highest probability score.

2.2 TAC KBP Slot Filling

Text Analysis Conference (TAC) is another workshop series which is sponsored by NIST[3]. It was developed to promote research in NLP and its affiliates by providing large scale test data, common evaluation procedures and forum for researchers to share their system results.

Similar to TREC, TAC also has a number of tracks such as Knowledge Based Population (KBP) track and Summarization track. In the KBP track, it focuses on developing a system that can detect whether an entity has already existed in the current knowledge bases, extract its detailed information from the data collection (e.g. web pages and newswire texts) and use this information to find/populate a new or existing knowledge base. Knowledge base is a special kind of database where important information is stored and it can be organized, shared, utilized by people. In TAC KBP, they use wikipedia infoboxes which provide an easy table format consisting of important facts that are related to the articles/topics. These facts are represented as a list of attributes (slots) with an answer (or filler) that corresponds to it.

For the slot filling task, they consider only PERSON and ORGANIZATION entities and from them, they extract all information that matches to the predefined slots. There are

[3]www.nist.gov/tac/about/index.html

25 slots for PERSON and 16 for ORGANIZATION defined in the task. Each slot is categorized based on the quantity and content of their fillers. More detailed information about TAC KBP Slot filling task can be found in Ellis [2013]. Example below illustrates the query in Slot filling task.

Example 5. *<query id="SF501">*
<name >Army Medical Command</name>
<docid >LTW_ENG_20080508.0012.LDC2009T13</docid>
<enttype>ORG</enttype>
<nodeid>E0388093</nodeid>
<ignore>org:city_of_headquarters org:country_of_headquarters
org:stateorprovince_of_headquarters</ignore>
</query>

Each query consists of an identifier, name, type (either PERSON or ORGANIZATION), a reference (*docid*) to a document in the corpus to disambiguate the query (e.g. if there are multiple entries with the same query name), *nodeid* (if the entity exists in the knowledge base) and list of attributes that should be ignored. In the above example, the query name is '*Army Medical Command*', and to find the information about it, the participant can check the *docid* that is mentioned there. It is considered as ORGANIZATION type and it has a reference to knowledge base with id *E0388093*. The task is to find all fillers for this query (16 slots) excluding three slots that are mentioned above.

From the 2012 Slot Filling task, we observed two submissions [Min et al., 2012, Roth et al., 2012] and found that both of them utilized pattern matching mechanism and classification tools in order to find the best slot. In addition, they also employed freebase[4] which is an open repository that records structured data consisting of 23 million entities.

From TREC and TAC, we can see the differences in three aspects:

1. Question format: in TREC, the question can be posed in form of natural language such as 'who is the father of Obama?', while in TAC, it is formalized in the form of query.

2. Entity type. in TREC, the type can varied from person, organization, location, substance, etc. and TAC only concerns with person and organization.

3. Answer type. in TREC, the questions are varied from single, list, definition, reasoning, etc. while in TAC, most of the time focus on single and list.

[4]http://www.freebase.com/

Figure 2.1: Akinator online game

2.3 Akinator

Akinator[5] is an interactive online game that is played between the user and the program. The game is built by Elokence.com which is an IT company located in France. This game is developed in 2007 and has received a lot of attention from users especially with the appearing of the mobile version for Apple and Android. A snapshot of the game is presented in Figure 2.1.

To start the game, first the user needs to think of a character either real or fictive (i.e. from a movie or cartoon, novel, etc). Next, the program, which is represented as the genie, will try to guest the character. To guest the character, the genie will prompt series of questions (around 20 questions) to the user. Each question can be answered with five options such as '*yes*', '*no*', '*don't know*', '*probably and probably not*'. The format of the question is basically a yes/no question, for example, '*Is your character famous because of youtube?*', '*Does your character have white skin?*', '*Was your character the first president of a country?*', and etc.

What makes this game interesting is that the user can submit a list of characters and short description about each of them to the game itself. Thus, the database for these characters is grown and updated. In addition, the user can add new questions to the game.

[5]http://en.akinator.com/

2.4 Jeopardy[6]

Another interactive game is Jeopardy, which is played in the U.S. since 1984. This games format involves three contestants playing against each other in a three rounds contest such as '*The Jeopardy!* and '*Double Jeopardy!* Rounds and '*Final Jeopardy!*.

The game is split into three rounds where each round has a different setting. In the first round, the contestant is provided with a grid of six columns. Each column contains a category and another 5 rows with the increasing dollar values (e.g. 200$, 400$, 600$, 800$ and 1000$). In the second round, the dollar values are doubled. In the final round, the contestant has to put a wager from 0 up to the maximum score which is already achieved from the first two rounds. If the contestant give a correct response, the amount of wager is added into his/her current score, otherwise it will be subtracted from the contestant's score.

The game is started when a previous winning contestant selects a category and the dollar value, for example '*President, 200*$'. After that, the host will present a clue to all contestants about the particular topic. Each contestant is provided with a handheld signaling button. The contestant who presses the button first has the opportunity to respond.

Unlike any other game where the contestant is questioned, in Jeopardy, the answer is given to the contestant. The contestant's task is to create a valid response by prompting a question. After the contestant select a category, the host prompts a cue '*The Father of Our Country, he didn't really chop down a cherry tree*'. The correct response that is expected from the contestant would be '*who is George Washington?*'.

In 2011, IBM with its system, which is called Watson [Ferrucci et al., 2010], won the Jeopardy game against the two other human contestants. This system is made for handling question answering that is posed in natural language. For this particular game, Watson has an access to dictionaries, encyclopedias, and any other text materials. Although it is not connected to the internet for this Jeopardy game, Watson still have 4 terabytes of storage which consists of structured and unstructured data including Wikipedia[7] as the information sources.

This system has a deep and long pipeline from content acquisition, question analysis, hypothesis generation, soft filtering, hypothesis & evidence scoring, final merging & ranking, answer merging, to ranking & confidence estimation. Inside of the hypothesis generation, it relies on named entity detection and also triple store and reverse dictionary lookup to

[6]Most of the information about the game is summarized from http://en.wikipedia.org/wiki/Jeopardy!
[7]http://www.wikipedia.org

generate candidate answers. After that a statistical approach is employed to measure the confidence score and filter the candidates answers.

2.5 Chapter Summary

In this chapter, we have shown the trend in the QA field. Some related works in TREC and TAC have proven that this field is still interesting for many researchers. In addition, real world applications such as Akinator and IBM Watson exist not only for the research community, but also for the public use.

The QA architecture that introduced by Moldovan et al. [2000] has been widely used and it serves as standard approach in many development of QA applications. Furthermore, the use of NE tags and pattern matching are beneficial in finding the correct answer. Moreover, real word applications such as Akinator give us valuable information about what kind of questions may be asked in order to guess the name of the famous person. Additionally, slot filling task describes which information is important to serve as the slot fillers for PERSON and ORGANIZATION type. From this task, we can assume that finding important biographical facts can be represented as finding correct slot fillers. [Min et al., 2012, Roth et al., 2012] emphasized the role of classification tools in refining the answer candidate. Last, we learn from Jeopardy that a more sophisticated QA system can be made by applying a hybrid approach that we mentioned in the previous chapter.

In the next chapter, we are going to explain the algorithms and tools that we used in our experiment.

Chapter 3

Algorithms and Tools

This chapter explains briefly the algorithms and concepts that are utilized to develop our Answer Extraction Module (AEM). In the next two sections, we describe the classification algorithm and pattern matching mechanism for our AEM. The last three sections cover the tools that we use for our experiment such as POS tagger, Chunker and NER tagger.

3.1 Classification

Before we go deeper into our classification mechanism, we start with the basic probabilistic models for classification task. There are two general models that are discussed here such as generative and discriminative models. Generative model learns the joint probability distribution and assumes that features are independent. Examples of this model are Naive Bayes, Hidden Markov Model (HMM). On the other hand, discriminative learns the conditional probability distribution which can be read as probability of the output (hidden variable) given the input (observation variable). Support Vector Machines (SVMs), Maximum Entropy (MaxEnt) and Conditional Random Fields (CRFs) are some examples of the discriminative model. Each model is described in more detail in the next section.

3.1.1 Generative Models

Given an input data $x \in X$, and a set of fixed-class $Y = y_1, y_2, \ldots, y_n$, the goal of classification task is predicting the correct class to the data $f : x \rightarrow y$. Using Naive Bayes classifier, this task can be formulated in below equation:

$$\hat{y} = \arg\max_{y \in Y} P(y|x) \tag{3.1}$$

The above equation means that out of all classes, we want the label that maximizes the probability of label y given data x. This $P(y|x)$ also can be represented using bayes rule as:

$$\begin{aligned} P(y|x) &= \frac{P(x|y).P(y)}{P(x)} \\ &= P(x|y).P(y) \end{aligned} \tag{3.2}$$

The denominator $P(x)$ can be dropped because it exists as a constant which will not affect the final result. The data x can be described as a set of input features, therefore $P(x|y)$ from equation 3.2 can be transformed into:

$$P(x|y) = P(x_1, x_2, \ldots, x_n|y) \tag{3.3}$$

where $x_1, x_2, \ldots, x_n$ are the features. The naive assumption that we used here is the features are independent, therefore we can also formulate the above equation into:

$$P(x_1, x_2, \ldots, x_n|y) = P(x_1|y).P(x_2|y).\ldots.P(x_n|y) \tag{3.4}$$

The final formulation of Nave Bayes can be formed by plugging Equation 3.2, 3.3 and 3.4 into Equation 3.1 which is described as follow:

$$\hat{y} = \arg\max_{y \in Y} P(y) \prod_{i=1}^{n} P(x_i|y) \tag{3.5}$$

This Naive bayes classifier works only with single variables. In our task we are working with sequence of data where the output of a certain state is influenced not only by its current state but also by the previous one. Thus, we need an extension of this Naive Bayes classifier which is Hidden Markov Model (HMM) [Rabiner, 1989]. HMM accommodates our needs with the Markov assumption that each future state only depends on the present state.

Given a sequence of input data $\overrightarrow{x} = (x_1, x_2, \ldots, x_n)$ and $\overrightarrow{y} = (y_1, y_2, \ldots, y_n)$ is the sequence class that corresponds to x, the formulation of finding y given x can be described as follow.

$$P(\overrightarrow{y}, \overrightarrow{x}) = \prod_{i=1}^{n} P(y_i|y_{i-1})P(x_i|y_i). \tag{3.6}$$

From Equation 3.6, the dependency between the current and previous state is shown in $P(y_i|y_{i-1})$ (transition probability). $P(x_i|y_i)$ is the emission probability where the observation at position i is determined by the hidden state at that position. In order to find the best path in HMM, we can apply Viterbi algorithm which is dynamic programming

algorithm to find the best sequence of hidden states.

Although HMM has shown promising results in many NLP applications, there are several drawbacks such as:

1. The independence assumption sometimes does not work in the real world where the features maybe dependent to each other.
2. It is hard to integrate features in the transition probability.

3.1.2 Discriminative Models

Another concept that is widely used is discriminative models. Using this model, the assumption is that the generation of the label (hidden variable) is derived from the data (observation). In the next two sections, we explain the use of discriminative model such as Maximum Entropy(MaxEnt) for single data and then continue with Conditional Random Fields(CRFs) for handing sequential data.

MaxEnt [Berger et al., 1996] is a conditional model that is based on the concept of entropy. With this concept, we aim to build such model that has fewest assumptions in the probability of distribution. In other terms, the best prediction is given by the data whose entropy is the highest.

Similar to section 3.1.1, Given an input data $x \in X$ and a set of fixed-class $Y = y_1, y_2, \ldots, y_n$ the MaxEnt model can be formulated as

$$P(y|x) = \frac{1}{Z(x)} \left(\exp \sum_{i=1}^{n} \lambda_i f_i(x_i, y_i) \right) \tag{3.7}$$

where $Z(x)$ is the normalization factor.

$$Z(x) = \sum_{y \in Y} \left(\exp \sum_{i=1}^{n} \lambda_i f_i(x_i, y_i) \right) \tag{3.8}$$

The features that are used to described the input data is put in f_i. Sometimes it is called as indicator function because it consists of only boolean representation (0 and 1). λ_i is the weight for the f_i. In this case, we want to select the weight so that the conditional maximum likelihood estimation can be achieved. In order to do that, certain optimization process is applied such as L-BFGS, Generalized Iterative Scaling (GIS) or Improved Iterative Scaling (IIS).

Conditional Random Fields (CRFs) [Lafferty et al., 2001] is an extension of MaxEnt to handle sequential data problem such as POS tagging or NER. This is a probabilistic

model that compute $\overrightarrow{y} = (y_1, \ldots, y_n)$ given a sequence of input like $\overrightarrow{x} = (x_1, \ldots, x_n)$. This model can be seen as a general interpretation of Hidden Markov Model and Maximum Entropy. In general CRFs can be formalized as

$$p(\overrightarrow{y}|\overrightarrow{x}) = \frac{1}{Z(\overrightarrow{x})} \prod_{j=1}^{n} \Psi_j(\overrightarrow{x}, \overrightarrow{y}) \tag{3.9}$$

where Ψ is a potential function defined as:

$$\Psi_j(\overrightarrow{x}, \overrightarrow{y}') = \exp(\sum_{i=1}^{m} \lambda_i f_i(y_{j-1}, y_j, \overrightarrow{x}, j)) \tag{3.10}$$

and $Z(\overrightarrow{x})$ is a normalization factor defined as:

$$Z(\overrightarrow{x}) = \sum_{\overrightarrow{y}'} \prod_{j=1}^{n} \Psi_j(\overrightarrow{x}, \overrightarrow{y}') \tag{3.11}$$

f_i is the transition/state function where we put the features that represent our data. Unlike HMM, in CRFs, we can put any feature to represent the connection between the current state and the previous one. In addition, the hidden variable is determined based on the observation variable. λ_i is the weight for the current data. To find the best weight that maximizes the likelihood of the training model we can use such as Limited-memory BFGS (L-BFGS), Quasi-Newton, Averaged Perceptron and many more.

Another model that learns sequences is SVM-HMM which combines the use of HMM and SVM [Joachims et al., 2009, Altun et al., 2003]. The benefit of HMM in this task is the Markov chain dependency and the efficient dynamic programming while applying SVM can help in adding overlapping features, and maximum margin principle and kernel approach to learn non-linear discriminant function.

Using SVM-HMM, this sequence classification task's goal is to maximize the discriminant function. Given an input sequence $\overrightarrow{x} = (x_1, \ldots, x_l)$, a sequence of output labels $\overrightarrow{y} = (y_1, \ldots, y_l)$, is predicted by the model according to the discriminant function as described below:

$$\overrightarrow{y} = \arg\max_{\overrightarrow{y}} \sum_{i=1}^{l} \left[\sum_{j=1}^{k} (x_i \cdot w_{y_{i-j},\ldots,y_i}) + \varphi_{trans}(y_{i-j}, \ldots, y_i) \cdot w_{trans} \right] \tag{3.12}$$

$w_{y_{i-j},\ldots,y_i}$ represents the emission weight vector for each different kth-order label sequence $y_{i-j}, \ldots, y_i$ while the transition weight vector between the states is represented as w_{trans}. $\varphi(y_{i-j}, \ldots, y_i)$ is used as the indicator vector corresponding to the sequence $y_{i-j}, \ldots, y_i$.

Given a set of training examples $(\overrightarrow{x}^1, \overrightarrow{y}^1), \cdots, (\overrightarrow{x}^n, \overrightarrow{y}^n)$ where $j-$training example is represented as $\overrightarrow{x}^j = (x_1^j, \ldots, x_l^j)$ with their correct label sequences $\overrightarrow{y}^j = (y_1^j, \ldots, y_l^j)$, the discriminant function in Equation 3.12 can be solved as optimization problem:

$$\min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{N}\xi_i \tag{3.13}$$

$$s.t. \forall \overrightarrow{y} : \left[\sum_{i=1}^{l}(x_1^1 \cdot w_{y_i^1}) + \varphi_{trans}(y_{i-1}^1, y_i^1) \cdot w_{trans}\right] \geq \left[\sum_{i=1}^{l}(x_1^1 \cdot w_{y_i}) + \varphi_{trans}(y_{i-1}, y_i) \cdot w_{trans}\right] + \Delta(y^1, y) - \xi_1$$

$$\cdots$$

$$\forall \overrightarrow{y} : \left[\sum_{i=1}^{l}(x_1^n \cdot w_{y_i^n}) + \varphi_{trans}(y_{i-1}^n, y_i^n) \cdot w_{trans}\right] \geq \left[\sum_{i=1}^{l}(x_1^n \cdot w_{y_i}) + \varphi_{trans}(y_{i-1}, y_i) \cdot w_{trans}\right] + \Delta(y^n, y) - \xi_n$$

where $\Delta(y^n, y)$ is the loss function or the number of labels that is misclassified in the sentence and C is the trades off between the margin size and training error.

3.2 Pattern matching

Regular expression is a sequence of characters that is used to create a search pattern. We can use a regular expression to find a string that matches the pattern. This idea is later called 'pattern matching'. Many programming languages such as Java, Python, Perl, Ruby, PHP, .net, etc. has implemented this regular expression in their code. For example, a regular expression *computer* will find any string in the text that contains *computer*.

Regular expression is case sensitive, so the above example will not match the string *COMPUTER*. Another example is *recogni[zs]e* can be used to find a string *recognize* or *recognise* in the text. The use of square bracket means that the pattern that we want to find has to match to one of the element inside of the square bracket (z or s). Another example *[a-z0-9]* will find any string that contains a lower case alphabet (from a-z) or a number between 0-9.

A caret symbol (^) can have its own meaning if it used in the regular expression. For example, a pattern *^He* matches with the string *He* (notice the capital H) and the word *He* has to appear in the front of the string. On the other hand, *$* symbol is used to match the end of a string. For example, *book$* pattern matches with *book* but not with *books*. In addition, carret symbol can have different meaning if it is used inside the square brackets. For example *[^a-z]* means that it negates any string that contains lower case letter.

Some range of numbers can be represented using regular expression. For this, there are several notation that can be employed such as:

1. * (kleene star) which means zero or more of the previous character. For example, xy*z matches the substring xz, xyz, xyyz, xyyyz, etc.

2. + (kleene +) which means one or more of the previous character. For example, a+bc matches the substring abc, aabc, aaabc, etc.

3. ? means zero or one of the previous character. For example, colou?r only matches to substring color or colour.

4. {n} means the n occurrences of the previous character. For example, hel{3}o matches with substring helllo (three l).

5. {n,m} means from n to m occurrences (inclusive) of the previous character. For example, hel{1,3}o matches with substring helo, hello, helllo.

6. {n,} means at least n occurrences of the previous character. For example, hel{1,}o matches with substring helo, hello, hellllllo, etc.

Moreover, there are some useful shortcuts in the regular expression [reference] such as:

1. \d means [0-9] or any digit

2. \D means [^0-9] or any non-digit

3. \w means [a-zA-Z0-9_] or any alphanumeric or underscore

4. \W means [^\w] a non-alphanumeric

5. \s means [\r\t\n\f] whitespace (space, carriage return, tab, line break, form feed)

6. \S means [^\s] non-whitespace

7. . means any single character except new line

Another important thing is, suppose we want to find a string contains either *gray* or *grey*, we can utilize vertical bar (|) to represent disjunctive operator. For example, *gray|grey* matches with either *gray* or *grey*. If we want to find a string *grumpy* or *grumpies*, we cannot use *grumpy|ies* because that would match only to *grumpy* or *ies* (the *y* character is bond with higher precedence with the previous characters). Therefore, we use the parenthesis in order to group the pattern. *grump(y|ies)* pattern matches with *grumpy* or *grumpies*. The disjunctive operator in this case only works inside of the parenthesis.

Continuing the parenthesis explanation, if we want to capture two information or strings from a particular pattern, we can use this parenthesis to get both of them. A pattern *born on (.*) at (.*)* will find any string that contains a phrase *born on* followed by any string and then the word *at* followed by any string again. An example that matches this pattern is *born on 29 December 1986 at Jakarta* where *29 December 1986* is captured by the first parenthesis (group 1) and *Jakarta* is captured by the second one (group 2). Section 5.2.3 expounds the use of pattern matching for our task.

3.3 Part-of-Speech Tagger

Part-of-Speech(POS) tagging is the process of marking each word in a text according to its POS. At the moment, there are many POS tagsets which have been developed for NLP communities. Since we work with English data and we employ Stanford POS tagger [Toutanova et al., 2003] which uses the Penn Treebank tagset Marcus et al. [1993]. This tagset contains 45 POS tags such as NN (noun), NNS (plural noun), VB (verb, base form), RB (adverb) etc. POS tagging is useful for many applications such as NER, word sense disambiguation, chunking, etc.

The output from Stanford POS tagger for the sentence '*This is a sample sentence*' can be seen below.

Example 6. *This/DT is/VBZ a/DT sample/NN sentence/NN*

In the above example, the word '*this*' and '*a*' are tagged as DT (determiner), '*is*' as VBZ (verb, 3rd person singular present), and the word '*sample*' and '*sentence*' are tagged as NN (noun).

For our purposes, this POS information is useful to know the difference category of the particular word. In addition, this tagger also captures the information such as PRP$ (possessive pronoun) which we use to hide the famous person's gender. Section 5.2.4 elaborates more about this POS tag in our work.

3.4 Chunker

Chunking is a term in the information extraction task that identifies and classifies flat segments (e.g. noun phrases, verb phrases, adjective phrases, prepositional phrase, etc.) of a sentence. Flat here means that the segment is not overlapping with others. The following example illustrates the output from openNLP chunker[1] for the sentence '*In 2007, Merkel was President of the European Council and chaired the G8*'.

Example 7. *[PP In] [NP 2007], [NP Merkel] [VP was] [NP President] [PP of] [NP the European Council] and [VP chaired] [NP the G8].*

In the above example, the sentence is divided into nine segments and each segment is identified by its phrase. For our task, this information is helpful especially to find the correct and complete answer for a given question. Section 5.2.1 discusses the use of chunk information for our task.

[1] http://opennlp.apache.org/index.html

3.5 Named Entity Recognition Tagger

Named Entity Recognition (NER) is an NLP task that focuses on detecting and classifying all proper names in a text. In the early times, NER tagger only concerns with finding the name of location, organization and person. Nowadays, there are plenty of named entity tags that can be recognized using NER tagger. For example, Stanford NER tagger [Finkel et al., 2005] can recognize seven entity types such as Time, Location, Organization, Person, Money, Percent, Date.

The standard procedure to find these NE types is by formatting it as sequence labeling task. This sequence labeling task uses IOB encoding information to mark the NE types. An example of using Stanford NER tagger in identifying Person and Location categories is shown in Table 3.1.

Text:	Gates	graduated	from	Lakeside	School	in	1973	.
Label:	PERSON	O	O	ORGANIZATION	ORGANIZATION	O	DATE	O

Table 3.1: Stanford NER tagger's output

In Table 3.1, Stanford NER tagger marks '*Gates*' as PERSON, '*Lakeside School*' as ORGANIZATION and '*1973*' as DATE. All of these tags are valuable to determine the answer type. Similar to Chunker, more detailed information about the use of NER tagger is covered in Section 5.2.1.

Chapter 4

Semantic Relations & Annotation

In this chapter we provide the concept of a semantic relation. The relations are used as the guideline to extract biographical facts about famous person. The definition that we have here are influenced by TAC KBP Slot filling task [Ellis, 2013]. In addition, the annotation definition and process are explained in this chapter.

4.1 Relations type

We define the semantic relation as a predicate with two arguments (person and entity, person and event, event and event). There are three types of semantic relations:

1. RELATION(Z,?X) describes the connection between Z which is the person in the question and X is the entity slot that needs to be filled. For example, CHILD_OF(EINSTEIN, ?X) means that we are looking for the child/children of Albert Einstein.

2. RELATION(E,?X) explains the connection between the event (E) in the question and X is the entity slot that needs to be filled. For example, DURATION(STUDY$_{Einstein}$, X) means that we are looking for study duration from Albert Einstein.

3. RELATION(E$_{1_z}$, ?E$_{2_z}$) describes the connection between Event 1 (E$_1$) in the question with the focus of the person in the question (Z) and Event 2 (E$_2$) is the event slot that needs to be filled with the focus of the same person (Z) as well. For example, REASON(DEATH$_{einstein}$, E$_{2_einstein}$).

Similar to TAC KBP Slot filling, each slot can be categorized based on its content into three types:

1. Name slots contains three NE categories:

(a) Person

This entity refers to individual human.

(b) Organization

This entity refers to organization, company, agency or group of people.

(c) Geo-political Entities

This entity refers to composite entities such as government, location, towns, cities, provinces, states and countries.

2. Value slots

These slots contain Date and Numerical value.

3. String slot

Everything that is not captured in the name and value slots is considered as string slot.

Besides the content, each slot also can be categorized based on the quantity:

1. Single value

One slot can have only a single filler. For example, a person can only have one location of birth.

2. List value

One slot can have multiple fillers. For example, a person may have multiple siblings.

4.2 Relations grouping

From the above types, we defined seven groups to cover the semantic relations:

1. Human descriptions
Properties that closely describe the person are classified into this group. Usually these properties are owned by the person since he/she is born. Name, age, weight, gender, nationality, title, religion and education are some properties that we consider to be part of this group.

2. Human relations
Every relation or connection that the person has with other humans is captured in this group. For instance, family relationships such as parent, child, and sibling are categorized into this group. In addition, friends and enemies are also part of this group.

3. Human groups
   This group consists of relations that occur between the person and other groups of people (e.g. organization, political party, etc.). If a person is a member of political party, or an owner/founder of a company, or works in some company is considered as part of this group. Furthermore, a person who becomes a victim of other group is belong to this group.

4. Entities
   This group comprises of the achievement or product that involves the person. The creation that the person made, the award that the person got are grouped here.

5. Location
   An event that took place in a certain location that involves the person is captured in this group. For example, the place where the person was born or spent whole life or even the place where the person died is part of this group

6. Time
   Similar to location group, this group covers the time information of the events that involves the person. Example such as time of birth or time of death of the person is suitable for this group.

7. Description
   Event descriptions such as a reason, a purpose or manner that involves the person, is captured in this group.

From these group, we defined 53 semantic relations that are summarized in the following table.

Group	**Relation**
Human descriptions	NAME, ALT_NAME, AGE_OF*, BODY, GENDER, NATIONALITY*, RELIGION*, TITLE*, EDUCATION_OF*
Human relations	CHILD_OF*, PARENT_OF*, SIBLING_OF*, SPOUSE_OF*, FAMILY_OF, FRIEND_OF, ENEMY_OF, COLLEAGUE_OF*, OTHER_REL
Human groups	MEMBER_OF*, OWNER_OF*, FOUNDER_OF*, EMPLOYEE_OF*, EMPLOYER_OF, SUPERIOR_OF, SUBORDINATE_OF*, SUPPORTER_OF*, SUPPORTEE_OF*, CHARGER_OF, CHARGEE_OF, CHARGED_FOR, CAUSE_OF, VICTIM_OF
Entities	CREATOR_OF*, AWARD*, ACCOMPLISHMENT*, ICON, ACTIVITY_OF, PART_IN*, ENT_OTHER
Location	LOC*, LOC_RESIDENCE*, LOC_BIRTH*, LOC_DEATH*
Time	TIME*, TIME_BIRTH*, TIME_DEATH*, DURATION*, PERIOD, FREQUENCY
Description	MANNER, PURPOSE, REASON, DEFINITION_OF

Table 4.1: Semantic Relations summary

For our experiment, we used only 29 relations (the one with $*$). We omitted the rest of the relations because they are less frequent, thus it is hard to train the classifiers. Moreover, we do not use NAME and ALT_NAME due to the game rule that prevents any direct question about the person's name.

4.3 Relations Mapping from TAC KBP Slot Filling Task

Some of the above relations are adopted from TAC KBP Slot Filling task. Table 4.2 displays the mapping of 23 TAC KBP slots into our semantic relations.

Type	**Slot Name**	**Relation**
person	age	AGE_OF
person	alternate_names	ALT_NAME
person	parents	CHILD_OF
person	employee_or_member_of	MEMBER_OF
		EMPLOYEE_OF
person	country_of_birth	LOC_BIRTH
person	stateorprovince_of_birth	
person	city_of_death	LOC_DEATH
person	country_of_death	
person	stateorprovince_of_death	
person	cities_of_residence	LOC_RESIDENCE
person	countries_of_residence	
person	statesorprovinces_of_residence	
person	schools_attended	EDUCATION_OF
person	origin	NATIONALITY
person	other_family	OTHER_FAMILY
person	children	PARENT_OF
person	cause_of_death	REASON
person	religion	RELIGION
person	siblings	SIBLING_OF
person	spouse	SPOUSE_OF
person	date_of_birth	TIME_BIRTH
person	date_of_death	TIME_DEATH
person	title	TITLE
person	charges	CHARGED_FOR
organization	founded_by	FOUNDER_OF

Table 4.2: TAC KBP Slots mapping

The first column in Table 4.2 describes the type of the first argument of the slot. In TAC KBP Slot filling task, they only have two types which are PERSON and ORGANIZATION. The second column explains the slot name that are defined in the task. The last column states the mapping between the TAC slots and our semantic relations.

4.4 Relations definition

In this section, we explain some of the relations that are not defined in TAC KBP Slot filling task.

1. ACCOMPLISHMENT(Z,X)

   Content: String

   Quantity: List

   Description: Persons, organizations, geopolitical entities which the assigned person has supported either economically, politically or morally.

Entity	Document passage	Correct Filler
Rick Warren	Warren received a Bachelor of Arts degree from California Baptist University in Riverside, California.	a Bachelor of Arts degree
Sigmeund Freud	Freud qualified as a Doctor of Medicine at the University of Vienna in 1881.	a Doctor of Medicine
Alfred Hitchcock	The magazine MovieMaker has described Alfred Hitchcock as the most influential filmmaker of all time.	the most influential filmmaker of all time

Table 4.3: ACCOMPLISHMENT examples

2. AWARD(Z,X)

   Content: Name or string

   Quantity: List

   Description: The entities which the assigned person has been awarded/honoured.

Entity	Document passage	Correct Filler
Willhelm Rontgen	In 1901 Rntgen was awarded the very first Nobel Prize in Physics.	the very first Nobel Prize in Physics
Gordon Ramsay	In July 2006, Ramsay won the Catey award for "Independent Restaurateur of the Year".	the Catey award for "Independent Restaurateur of the Year".
Lady Gaga	Her achievements include five Grammy Awards and 13 MTV Video Music Awards.	five Grammy Awards, 13 MTV Video Music Awards

Table 4.4: AWARD examples

3. COLLEAGUE_OF(Z,X)

   Content: Name

   Quantity: List

   Description: people/organization that are in employment or teamwork relation to the assigned person.

Entity	Document passage	Correct Filler
Nelson Mandela	In 1993, he received the joint Nobel Peace Prize with de Klerk.	de Klerk
Oprah Winfrey	In 1985, Winfrey co-starred in Steven Spielberg's The Color Purple as distraught housewife, Sofia.	Steven Spielberg
Jay-Z	Jay-Z collaborated with M.I.A. on the single "XXXO".	M.I.A

Table 4.5: COLLEAGUE_OF examples

4. CREATOR_OF(Z,X)
   Content: Name
   Quantity: List
   Description: The entities which the assigned person has created/produced that has not existed before this particular activity/event.

Entity	Document passage	Correct Filler
Lady Gaga	Lady Gaga came to prominence as a recording artist following the release of her debut album, The Fame (2008).	The Fame
Steve Jobs	Through Apple, he was widely recognized as a charismatic pioneer of the personal computer revolution.	the personal computer revolution
Johannes Kepler	Johannes Kepler's first major astronomical work, Mysterium Cosmographicum, was the first published defense of the Copernican system.	Mysterium Cosmographicum

Table 4.6: CREATOR_OF examples

5. DURATION(E,X)
   Content: Value
   Quantity: Single
   Description: Length of time for the event exists or lasts.

Entity	Document passage	Correct Filler
George W. Bush	Bush attended Yale University from 1964 to 1968, graduating with an B.A. in history.	from 1964 to 1968
Lance Armstrong	Armstrong had won the Tour de France a record seven consecutive times between 1999 and 2005.	between 1999 and 2005
Leonardo da Vinci	Leonardo worked in Milan from 1482 until 1499.	from 1482 until 1499

Table 4.7: DURATION examples

6. FOUNDER_OF(Z,X)
   Content: Name
   Quantity: List
   Description: The organization(s) or geopolitical entities (countries, governments) that the assigned person has formed, build and/or founded.

Entity	Document passage	Correct Filler
Steve Jobs	He is best known as the co-founder, chairman, and CEO of Apple Inc.	Apple Inc
Alexander Graham Bell	in 1888 , bell became one of the founding members of the National Geographic Society.	National Geographic Society
Genghis Khan	present-day Mongolians regard him as the founding father of Mongolia.	Mongolia

Table 4.8: FOUNDER_OF examples

7. LOC(E,X)
   Content: Name
   Quantity: Single
   Description: all other places which are not specified in LOC_RESIDENCE, LOC_BIRTH, LOC_DEATH.

Entity	Document passage	Correct Filler
Adam Smith	In 1778, Smith was appointed to a post as commissioner of customs in Scotland.	Scotland
Park Geun-hye	Park received honorary doctoral degrees from the Chinese Culture University, in Taiwan in 1987.	Taiwan
Max Planck	In Berlin, Planck joined the local Physical Society.	Berlin

Table 4.9: LOC examples

8. OWNER_OF(Z,X)
   Content: Name or value or string
   Quantity: List
   Description: The organization(s) or other entities that are the property of the assigned person.

   (a) Companies, money, residencies, housings are considered as the property

Entity	Document passage	Correct Filler
Steve Jobs	In 1986, he acquired the computer graphics division of Lucasfilm.	the computer graphics division of Lucasfilm
Tiger Woods	According to Golf Digest Tiger Woods made $769,440,709 from 1996 to 2007.	$769,440,709
Bill Gates	Among Gates's private acquisitions is the Codex Leicester, a collection of writings by Leonardo da Vinci.	the codex leicester

Table 4.10: OWNER_OF examples

9. PART_IN(Z,X)
   Content: Name or string
   Quantity: List
   Description: Activity that includes the assigned person as one of the participants.

10. SUBORDINATE_OF(Z,X)
    Content: Name
    Quantity: List
    Description: The person(s) who is in a position of less power or authority than someone else.

Entity	Document passage	Correct Filler
Melinda Gates	She was project manager for Microsoft Bob, Microsoft Encarta and Expedia.	Microsoft Bob, Microsoft Encarta and Expedia
Steve Jobs	As the new CEO of the company, Jobs oversaw the development of the iMac, iTunes, iPod, iPhone, and iPad, and on the services side, the company's Apple Retail	the iMac, iTunes, iPod, iPhone, and iPad
Mehmet Oz	Oz appeared as a health expert on The Oprah Winfrey Show for five seasons.	The Oprah Winfrey Show

Table 4.11: PART_IN examples

(a) Note that both verbs (e.g., oversees, commanded, leads, etc.) and titles (CFO, CEO, President, Vice president, director, etc.) can be used as justification for selecting subordinate fillers.

(b) Former subordinate(-s) are acceptable fillers.

Entity	Document passage	Correct Filler
Alexander The Great	At age 16, Alexander's education under Aristotle ended.	Aristotle
Kevin Rudd	Returning to Australia in 1988, he was appointed Chief of Staff to the Opposition Leader in Queensland, Wayne Goss.	Wayne Goss
Rihanna	Rihanna has worked with music video director Anthony Mandler on more than a dozen music videos.	Anthony Mandler

Table 4.12: SUBORDINATE_OF examples

11. SUPPORTER_OF(Z,X)
Content: Name or string
Quantity: List
Description: Persons, organizations, geopolitical entities which the assigned person has supported either economically, politically or morally.

(a) Former supported organizations of the assigned entity are valid fillers.

(b) If it is clear that either an Member of or Supporter relationship exists between the entity and the assigned person, but it is unclear which of the two slots correctly defines the relationship, Member of should be selected

Entity	Document passage	Correct Filler
J.K. Rowling	Rowling is a supporter of The Shannon Trust.	The Shannon Trust
Larry Page	Page is an investor in Tesla Motors.	Tesla Motors
Michael Bloomberg	Bloomberg reports giving $254 million in 2009 to almost 1,400 nonprofit organizations.	1,400 nonprofit organizations

Table 4.13: SUPPORTER_OF examples

12. SUPPORTEE_OF(Z,X)
    Content: Name or string
    Quantity: List
    Description: persons, organizations, geopolitical entities by which the assigned person has been supported either economically, politically or morally (the inverse of SUPPORTER_OF).

    (a) While the term supportee is most commonly used in reference to commercial organizations, the slot is meant to include other types of organizations as well. Correct fillers for this slot include regional branches of a central organization, organization departments or sports teams of a university.

    (b) A brand is generally not considered a supportee.

    (c) Former sponsor organizations are acceptable responses.

Entity	**Document passage**	**Correct Filler**
Mark Zuckerberg	Since 2010, Zuckerberg has been named among the 100 wealthiest and most influential people in the world by Time magazine.	Time magazine
Angela Merkel	On 23 May 2013, she was awarded an honorary doctorate from the Radboud University Nijmegen.	the Radboud University Nijmegen
Sonia Gandhi	Sonia was listed as one of the fifty best-dressed over 50s by the Guardian in March 2013.	The Guardian

Table 4.14: SUPPORTEE_OF examples

13. TIME(E,X)
    Content: Value
    Quantity: Single
    Description: any other times which are not specified in TIME_BIRTH and TIME_DEATH.

Entity	**Document passage**	**Correct Filler**
Franz Josef	On 24 April 1854 he married Elisabeth Amalie Eugenie von Wittelsbach (aka Sisi).	24 April 1854
Napoleon Bonaparte	In 1810 Napoleon married Archduchess Marie Louise of Austria.	1810
Hillary Clinton	She embarked on a career in law after receiving her J.D. from Yale Law School in 1973.	1973

Table 4.15: TIME examples

The rest of the relations is defined in Appendix A.

4.5 Annotation Guidelines

After the relations are defined, the next thing to do is identify them in the data that we have. To do so, we followed the guidelines as:

1. Both definite and indefinite NPs should be marked
2. All modifiers should be included in the markable such as:
   (a) definite and indefinite articles (e.g. '*the president of USA*', '*an engineer*')
   (b) Quantifiers such as '*many*', '*much*', '*several*', etc.
   (c) Text inside the brackets (but exclude the brackets).
   (d) Comma should be excluded except for date (e.g. '*August 4, 1985*')
   (e) Adjective modifiers (e.g. '*The richest man*')
   (f) Compounds (e.g. '*football player*')
3. Coordinated NP are treated separately and excluding coordination word and commas except for '*or*' (e.g. '*He is an athlete, scientist and teacher*')
4. Numerals, values, dates and quantifications (e.g. ' *$300 million*')
5. Exclude possessive pronouns appears in the beginning of NPs

4.6 Relations Labeling

Based on the above semantic relations definition and annotation guidelines, we start labeling the word tokens in the data. Table 4.16 illustrates our labeling process.

Text:	Alfred	Hitchcock	was	an	English	film	director	and	producer	.
Label:	O	O	O	O	B-NATIONALITY	B-TITLE	I-TITLE	O	B-TITLE	O

Table 4.16: Label using IOB-prefix

From Table 4.16, to mark the chunks, we use IOB-prefix (Inside, Outside, Beginning) and then followed by the relation name. Our rules only allow that B-prefix must appear before the I-prefix for the same relation and also B-prefix is not allowed to appear after the I-prefix for the same relation. In addition, I-prefix is not allowed to appear after O-prefix.

When we started the labeling process, we encountered some tricky cases. For example:

Example 8. *Once there, he became a favorite student of [Professor [August Kundt]].*

The chunk '*Professor August Kundt*' or just '*August Kundt*' is suitable for SUBORDINATE_OF relation. Both of them are correct because they contain the professor's name. However, we need to decide which one we will use to evaluate the system's performance. In the end, we selected the first option because it has broader coverage. If the system output only predicted the person's name, we could still expand it based on the NP boundaries. On the contrary, if we chose the latter option as the reference and the system predicted the whole chunk as the output, it would be more difficult to reduce the chunk coverage.

Example 9. *Alfred Hitchcock was [an [English] [film director]] and producer.*

In Example 9, '*an English film director*' can be marked as TITLE. However, there are two important information contained in this chunk which are the nationality of the person and his title. If we mark the whole chunk as TITLE, we will loose the information about the nationality and it is possible that the other sentences may not have this information. Therefore, we split this chunk into two parts such as '*English*' (NATIONALITY) and '*film director* (TITLE).

Example 10. *on [27 march 1785] , Marie Antoinette gave birth to a second son, Louis Charles , who was created the duc de Normandie.*

In addition to the previous example, in Example 10 the chunk '*27 march 1785*' is possible to have as TIME and TIME_BIRTH labels. However, the main character is Marie Antoinette, the TIME_BIRTH relation only applies to her birthday and not for the others. Therefore, we marked that chunk with TIME relation.

Chapter 5

System Architecture

5.1 Data and Preprocessing

In the beginning of our work, we were provided with 18 descriptions, each consisting of approximately 16 sentences with average length of 12 words. These descriptions, however, were not enough to train the classifiers to achieve good performance. Thus, we took an initiative to collect more descriptions from Wikipedia. The reasons we used Wikipedia are that this website provides a lot of description of famous people from around the world and these descriptions are formatted as raw texts.

We selected data from Wikipedia manually and did some preprocessing of it. The current set has 100 descriptions from those 71 describe male and 29 female persons; 51 people of these descriptions have deceased and the rest are still alive until this report is written.

Some preprocessing steps were performed in the beginning. First, the number of abbreviations needed to be reduced in order to minimize the tokenization problem for NER taggers since they had their own tokenizing mechanism. Second, long sentences had to be shortened because the focus of the sentence had often shifted from the main person to the others (e.g. his/her mother or father).

After the preprocessing was done, we labeled the descriptions with the semantic relations that we defined in Section 4.2. In total, we have 3988 instances that are marked. TIME relation holds the highest frequency with 579 instances followed by TITLE, CREATOR_OF. The distribution of the relations can be seen in Table 5.1.

RELATION	#	RELATION	#	RELATION	#	RELATION	#	RELATION	#
ACCOMPLISHMENT	158	DURATION	71	LOC_DEATH†	32	PART_IN	143	SUPPORTER_OF	33
AGE_OF†	84	EDUCATION_OF†	167	LOC_RESIDENCE†	127	RELIGION†	26	TIME	579
AWARD	100	EMPLOYEE_OF†	87	MEMBER_OF†	73	SIBLING_OF†	91	TIME_BIRTH†	112
CHILD_OF†	143	FOUNDER_OF†	46	NATIONALITY†	124	SPOUSE_OF†	76	TIME_DEATH†	41
COLLEAGUE_OF	67	LOC	223	OWNER_OF	44	SUBORDINATE_OF	52	TITLE†	562
CREATOR_OF	338	LOC_BIRTH†	196	PARENT_OF†	148	SUPPORTEE_OF	45		

Table 5.1: List of defined semantic relations.
† means that the relation is adopted from TAC KBP slot filling task.

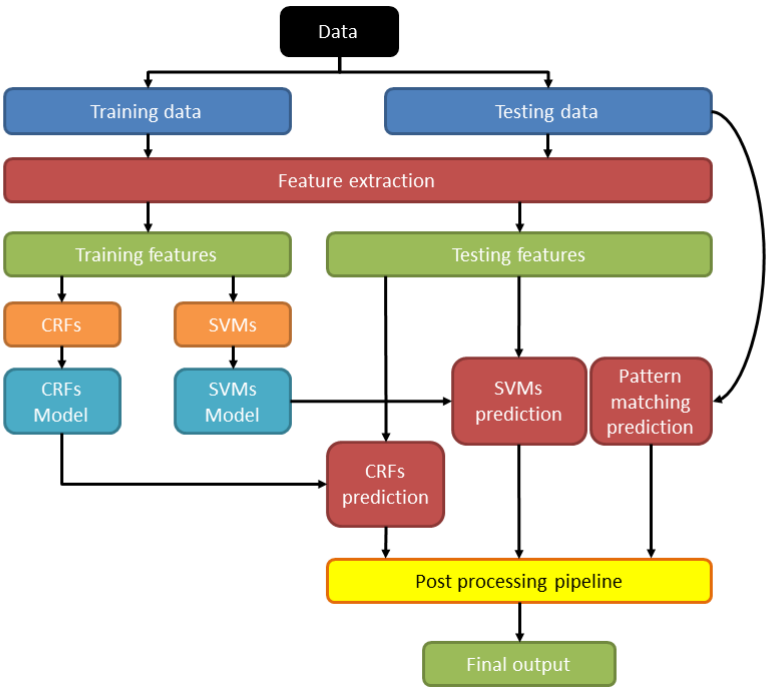

Figure 5.1: Answer extraction module architecture

5.2 Answer Extraction Module

Figure 5.1 displays our AEM architecture. First, the module read the data and then split it into training (80%) and testing set (20%). More explanation about data partition is available in Section 6. Furthermore, feature extractors were applied to each dataset to extract important features such as word, lemma, POS tag, NE tag and many more. Next section will provide the detailed description of these features. Moreover, two classifiers (CRFs and SVMs) are trained on these features to create the learning models and use them to predict the relation label for the testing data. Subsequently, pattern matching tool was employed to predict the relation labels. Finally, post processing mechanism was applied to filter the prediction output from the classifiers and pattern matching tool.

5.2.1 Feature Extraction

There were six features extracted from the data in order to train the classifiers. These features are:

1. **Word and lemma tokens**
   The word and lemma were extracted using the Stanford tokenizer and lemmatizer using Stanford CoreNLP toolkit[1].

2. **POS tags**
   In addition to the first feature, POS information was employed to disambiguate the word and lemma. We used Stanford POS tagger [Toutanova et al., 2003] to extract POS.

3. **Chunk**
   As already mentioned earlier in section 3.4, chunk information was extracted using OpenNLP toolkit[2].

4. **Named Entity tags**
   NER taggers such as Stanford NER [Finkel et al., 2005], Illinois NER [Ratinov and Roth, 2009], and Saarland NER [Chrupala and Klakow, 2010] were employed to detect NE in the data. Stanford NER tagger provided 7 categories such as Time, Location, Organization, Person, Money, Percent, Date. In addition, Illinois NER tagger covered 18 categories such as Nationality, Geo-Political Entity, Language, Work of Art, and many more. Finally, Saarland NER tagger identified 17 categories including Job-Title, Cause-Death, Religion, etc.

5. **Capitalization**
   This feature gave us boolean information if the first character of the word is in upper case or not. In addition, it replaced all upper case letter from the word and lemma into lower case. There are two reasons why we use this feature. First, we wanted to reduce the sparseness of the data. Many words have the same meaning, but they are considered to be different because one may appear at the beginning of the sentence (with upper case) while the others appear at the middle or end of the sentence. Second, this information can fill the gap of the NE tag if the NER tagger failed to detect it.

6. **Keywords**
   To assist the classifiers in predicting the correct label in the sentence, we chose a number of words (usually noun or verb) that are good indicators for certain relation

[1] http://nlp.stanford.edu/software/corenlp.shtml
[2] http://opennlp.apache.org/

in the sentence. Unlike the above features that were extracted automatically, this feature was manually selected based on our observation. We create cluster of keywords as a marker to certain relation. For example, words such as *marry*, *marriage*, *married*, *husband*, *wife*, *widow*, and *spouse* are grouped into one cluster which is SPOUSE_OF. Thus, every time there is a sentence that contains one of these keywords, we mark the sentence with additional flag. The full list of the keywords and their clusters can be seen in Table 5.2.

Relation	Keyword
AGE_OF	age, aged
AWARD	award, nobel, prize, grammy, scholarship, medal
ACCOMPLISHMENT	master, bachelor, phd, honorary, honoris, doctor, degree, rank, forbes, ph.d., champion, doctorate, top, greatest, richest, wealthiest, smartest, strongest, mba, b.s., m.b.a, m.sc., msc, m.sci., m.si., sc.m., m.s., mshs, ms, mag., mg., mgr, s.m., sm, m.a., ma, med, msc, msw, mpa, mph, mca, mla, alm, mba tech, mcom, mbus, psm, meng, mim, mem
CHILD_OF	father, mother, son, daughter, children, child, parent
COLLEAGUE_OF	with, behind, ahead, alongside, meet, visiting, visit, co-star, assist, team, include
CREATOR_OF	write, introduce, announce, author, originator, establish, autobiography, theory, compose, complete, devise, formulate, founder, pioneer, pioneering, direct, director, debut, release, formalize, publish, discover, creation, invention, album, create, founding, know, produce, papers, essay
DURATION	between, from, for, year, decade, during, after
EDUCATION_OF	enrol, study, learn, graduate, attend, alumnus, educate, education, scholarship, pupil, master, bachelor, phd, university, college, school, institute, student, class
EMPLOYEE_OF	work, contract, serve, become, teach, wrote, position, lecture, teacher, teaching, career, intern, internship, company, recruit, professor, hire, employee, research, apprenticeship
FOUNDER_OF	cofounder, co-founder, found, founder, founding, cofound, co-found, establish, launch, form, co-creator, create
LOC_BIRTH	birthplace, born, bear, birth
LOC_DEATH	die, assassinate, kill, murder, death, behead
LOC_RESIDENCE	childhood, live, left, leave, return, move, go, raise, home, house, migrate, emigrate, transmigrate, imigrate, grow, relocate, spend, stay, retire
MEMBER_OF	team, member, youth, squad, charity, party, club, senate, gild, society, leader, guild, union, parliament, association, congress, movement, committee, association, council, army, alliance
NATIONALITY	citizen, citizenship, nationality, national
OWNER_OF	donate, wealth, donation, invest, earn, own, possess, retain, hold, acquisition, acquire, worth, buy, give, salary
PARENT_OF	children, child, twin, daughter, son, baby, boy, girl
PART_IN	film, appear, role, career, play, music, military, movement, development, oversee, participate, collaborate, win, involve, direct, influential, assist, feature
RELIGION	buddhism, buddhist, confucianism, confucianist, hinduism, hinduist, islam, islmist, judaism, judaist, protestantism, protestant, catholicism, catholic, shinto, taoism, taoist, baha'i, christianity, christian, jainism, sikhism, zoroastrianism, christianity, secular, nonreligious, agnostic, atheist, primal-indigenous, sikhism, juche, spiritism, cao dai, tenrikyo, neo-paganism, unitarian-universalism, rastafarianism, scientology, methodist, muslim, anglican
SIBLING_OF	sibling, sister, brother, half-brother, half-sister
SPOUSE_OF	married, husband, wife, marry, marriage, spouse, widow
SUBORDINATE_OF	under, teacher, learn, advisor, adviser, supervision, teach, supervisor, tutor, student, tutelage, educate, direct, successor, mentore, mentor, disciple, assistant, apprentice
SUPPORTEE_OF	award, rank, honorary, greatest, top, wealthiest, richest, smartest, receive, powerful, list, world, honoris
SUPPORTER_OF	support, supporter, donate, give, campaign, patron, patronize, invest, investor, endorse, provide, help
TIME_BIRTH	birthplace, born, bear, birth
TIME_DEATH	die, assassinate, kill, murder, death, behead

Table 5.2: List of keywords

Along with marking good candidate sentences, the keyword is also used to mark sentences that are not supposed to have a particular relation (we called it as non-

keyword). For example, a sentence that has keywords such as *honorary*, *professor* are unlikely to have particular EDUCATION_OF relation. The list of non-keywords can be seen in Table 5.3.

Relation	Non-Keyword
EDUCATION_OF	honorary, professor, work, direct deliver, teaching, build, establish
EMPLOYEE_OF	member
NATIONALITY	fluent, proficient
SUPPORTEE_OF	undergraduate, bachelor, master,phd, ph.d.

Table 5.3: List of non-keywords

Apart from the above mentioned features, we tried dependency parsing information as the classifiers' feature but our preliminary work showed that this feature reduced the prediction quality. We believe that this lower quality is caused due to many dependency labels which lead to sparse data and it is noisy to be used as a feature.

SID	WID	Word	Lemma	POS	Stanford NER	Chunker	Illinois NER	Saarland NER	Keyword	Capitalization	Label
0	0	albert	albert	NNP	PERSON	B-NP	PERSON-B	B-PERSON	KEY_TIME_BIRTH	TRUE	O
0	1	einstein	einstein	NNP	PERSON	I-NP	PERSON-I	I-PERSON	KEY_TIME_BIRTH	TRUE	O
0	2	was	be	VBD	O	B-VP	O	O	KEY_TIME_BIRTH	FALSE	O
0	3	born	bear	VBN	O	I-VP	O	O	KEY_TIME_BIRTH	FALSE	O
0	4	on	on	IN	O	B-PP	O	O	KEY_TIME_BIRTH	FALSE	O
0	5	march	march	NNP	DATE	B-NP	DATE-B	B-DATE	KEY_TIME_BIRTH	TRUE	B-TIME_BIRTH
0	6	14th	14th	JJ	DATE	I-NP	DATE-I	I-DATE	KEY_TIME_BIRTH	FALSE	I-TIME_BIRTH
0	7	1879	1879	NNS	DATE	I-NP	DATE-I	I-DATE	KEY_TIME_BIRTH	FALSE	I-TIME_BIRTH
0	8	in	in	IN	O	B-PP	O	O	KEY_TIME_BIRTH	FALSE	O
0	9	ulm	ulm	NNP	LOCATION	B-NP	GPE-B	B-GPE:CITY	KEY_TIME_BIRTH	TRUE	B-LOC_BIRTH
0	10	,	,	,	O	O	O	O	KEY_TIME_BIRTH	FALSE	O
0	11	germany	germany	NNP	LOCATION	B-NP	GPE-B	B-GPE:COUNTRY	KEY_TIME_BIRTH	TRUE	B-LOC_BIRTH
0	12	.	.	.	O	O	O	O	KEY_TIME_BIRTH	FALSE	O

Table 5.4: Feature Extraction and Label annotation

Table 5.4 provides a brief example on how the extraction and labeling is done in the text. The column SID describes the sentence *id* for this particular sentence. It starts with the index 0. The next column contains word *id* for each word in the sentence. The third and fourth columns present the word and lemma respectively in lowercased form. In addition, the POS tag is available the fifth column. NE tag is provided in the sixth, eight, ninth column, while the chunking information is described in the seventh column. Since the sentence contains the word *born* which we assume is a good feature to predict the TIME_BIRTH relation, we fed this information (i.e 'KEY_TIME_BIRTH') to the classifiers. The next column captures boolean information whether the word in the sentence is written in upper case or not. Finally, the last column contains a relation tag consisting of relation label and IOB prefix.

5.2.2 Classifiers

Two classifiers such as SVMs and CRFs were trained using the extracted features described in the previous section. For the SVMs implementation, we use SVM^{hmm} package[3],

[3]http://www.cs.cornell.edu/people/tj/svm_light/svm_hmm.html

which supports sequence classification. In our preliminary work, we try kernel functions such as linear, polynomial, radial and sigmoid but the fastest one is linear[4]. In addition, we also set the structural learning algorithm such as 1-slack algorithm (primal) and 1-slack algorithm (dual). The best result is achieved with the latter one. We set the epsilon (precision of the classifiers) up to 0.001. The smaller epsilon, the longer the process and the more memory is needed for the training but the solution is more precise. Last, we set c parameter (trading-off slack vs. magnitude of the weight-vector) to 100. Our preliminary results show that bigger c increase the prediction quality.

In addition to SVMs, we employed two CRFs packages such as CRF++[5] and CRFsuite [Okazaki, 2007]. The differences between these two is that CRFsuite provides more training methods such as Averaged perceptron (AP), Limited-memory BFGS (LBFGS), Stochastic Gradient Descent, and etc. However, due to time constraints, not all of algorithms were tested. Nevertheless, we tried LBFGS and AP which worked pretty well in our preliminary experiment.

5.2.3 Pattern Matching

In addition to the classifiers, we developed a number of regular expressions to predict the relation label. These regular expressions were manually handcrafted for 12 relations such as CHILD_OF, AGE_OF, DURATION, etc. A list of regular expressions for DURATION can be found in Table 5.5.

Regular Expression	Group No.	Relation
^(from (([0-9]+) (to|until|through) ([0-9]+)))	1	DURATION
\s(from (([0-9]+) (to|until|through) ([0-9]+)))	1	DURATION
^(between (([0-9]+) and ([0-9]+)))	1	DURATION
\s(between (([0-9]+) and ([0-9]+)))	1	DURATION
^(for|after|lasted|during|spanning|over) ((.*) (month|months|year|years|decade|decades|seasons)) ([\.,\/#!$%\^*;:{}=\-_~()]|\s)	2	DURATION
\s(for|after|lasted|during|spanning|over) ((.*) (month|months|year|years|decade|decades|seasons)) ([\.,\/#!$%\^*;:{}=\-_~()]|\s)	2	DURATION

Table 5.5: DURATION regular expressions

In the table above, there are 3 columns defined. The first column describes the regular expression for the relation that is defined in the third column. The second column consists of the group number that we want to lookup for finding the relevant string. For example, *^(between (([0-9]+) and ([0-9]+)))* means find any string that starts with the word *between* followed by any number (at least 1 digit) and then the word *and* and then with another number (at least 1 digit). Since we use parenthesis here, the string that we want to capture is available in group 1. The full list of regular expressions for 11 relations is available in Appendix C.

[4]The others kernels are too slow and this also confirmed by the developers of the package.
[5]https://code.google.com/p/crfpp/

After the information is captured using regular expressions, we finalize the output by applying NER taggers to confirm the NE types. For example, regular expression like '*he married to (.*)*' should capture the information about the spouse of the person. This information checked by NER tagger to confirm whether it is a PERSON type or not.

5.2.4 Post processing pipeline

The process to predict the relation label did not stop with utilizing classifiers and pattern matching tool. The outputs from both tools were checked in order to select the best result for each relation. This checking involved four procedures such as possessive pronoun removal, overlap relation removal, non-argument-1 deletion and chunking expansion.

In the first procedure, all possessive pronouns that appear in the beginning of the relation label were removed in order to hide the gender information from the user. This is done to make the game more interesting. For example:

Example 11. *He and [his three siblings] were raised by their mother, Gloria Carter, after their father abandoned the family.*

In Example 11, the classifiers assign '*his three siblings*' with SIBLING_OF. Using this procedure, we eliminate '*his*' from the chunk label either by checking its POS or the word token (e.g. '*his*', 'her')

Second, sometimes the classifiers can predict more than one label of a certain chunk. These labels, however, are not acceptable due to our labeling standard, which only allow each chunk to have one label. Therefore, we had to remove one of the labels that was assigned to the chunk.

Example 12. *He was raised in [Ukiah], [California], and graduated from Ukiah High School in 1972.*

Both '*Ukiah*' and '*California*' in Example 12 were marked by the classifiers as LOC and LOC_RESIDENCE. However, LOC_RESIDENCE was selected as the final label because it has more specific information (LOC_RESIDENCE is subset of LOC). Besides this specific information, we can also set the selection process based on the priority. Usually this priority is set by some confidence score (in this case we use F-score). The higher the F-score the higher the priority.

Another procedure consists of non-argument-1 deletion. We omit every label that is given to a chunk whose focus is not the main person in the text. To get clearer point of view, please see the example below.

Example 13. *Her mother, Kathy Hilton is a [socialite] and [former actress], and her father, Richard Howard 'Rick' Hilton, is a [businessman].*

The classifiers label '*socialite*', '*former actress*' and '*businessman*' as TITLE. However, if we look carefully, this TITLE is referring to someone else which is either mother or father of the person. Therefore, we omitted every TITLE that appeared in that sentence.

Last, chunk expansion is employed for labels like TITLE and SUBORDINATE_OF. For TITLE, we expand the chunk if the next word after the chunk is '*of*' and followed by ORGANIZATION entity tag.

Example 14. *She later became the president of property developer SC Asset and* ***managing director*** *of Advanced Info Service.*

The classifiers marked '*managing director*' as TITLE from above example, while in our reference, the correct one is '*managing director of Advanced Info Service*'. The classifier missed this because usually it stopped at the end of NP and the PP information from the chunker only covered the preposition itself. Therefore, we expand the chunk in order to cover the full NP. Similar to TITLE, in SUBORDINATE_OF, if the classifiers only assign the person's name then we need to expand the NP boundaries so it covers the title of the person as well.

Chapter 6

Experiment Setup

We mentioned earlier that we split the data into training and testing set. The way we split the data was by using 5-fold cross validation procedure. We split the data into 5 partitions so that each partition consists of 20 biographies. Each partition is used as testing set and the remaining will be considered as training set. This procedure was repeated five times until each partition is used as testing set. By doing this, we can assess the system's performance on the new data, but also reduce the chance of overfitting in the experiment.

One may ask whether the proportion of each relation is equally distributed in each partition. Our answer is that it is hard to maintain this distribution because one sentence may contain multiple instances of the relation while the total number of relation instances differs from one to another. Therefore, the best solution is to split the data based on the number of biographies that we have.

There were two experiment sets prepared. In the first experiment, we investigated whether the classifiers that are trained separately for each relation can outperform the classifiers that are trained for all relations at once. Our intuition said that the former one can perform better due to the flexibility to select the features for each relation. In addition, we liked to know if the pattern matching is better than the classifiers. Finally, we wanted to know if the post processing pipeline can help to filter the output from the classifiers and pattern matching. Section 6.1 covers more detailed information about this experiment

In the second set, we examined the classifiers learnability given different size of the training set. Further explanation about the second set of experiment can be found in Section 6.2.

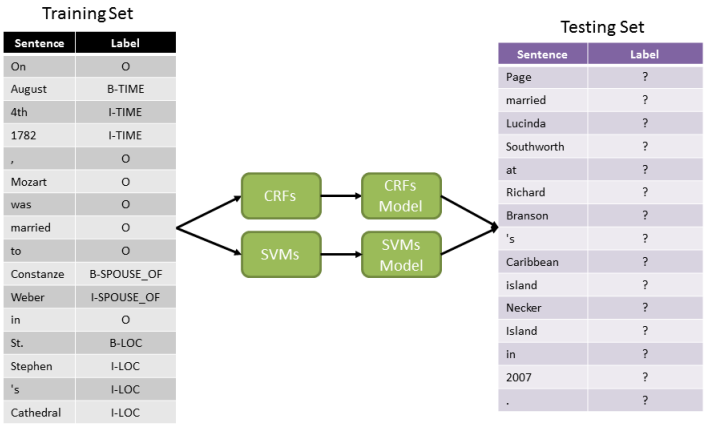

Figure 6.1: Classifiers for the baseline

6.1 Experiment 1

We designed three systems for our first set of experiment, which were the Baseline, System 1 and System 2. As a baseline, it is common to use the majority class as the final output. However, the most frequent instances that we have is TIME (579 instances) and to use that as the majority class will result in 579/3988 or equal to 14.52% accuracy. This number is not informative and too easy to be outperformed. Therefore, we set our baseline by training the classifiers (CRFs and SVMs) on automatically derived features and use these to predict all semantic relations at once. The features that we use to train the classifiers are coming from the NLP tools that we mentioned earlier. Features such as word and lemma tokens, POS tags, NE tags, chunk, and capitalization information (without any keyword) are extracted up to 3-grams before and after the current position. Figure 6.1 illustrates how the baseline works.

For System 1, we trained the classifiers to learn the model for each relation separately. Figure 6.2 illustrates the scheme for System 1. First, we split the training set according to each relation and set the label for other relations into *O*. Second, we extract the features for each relation and train the classifiers. In addition to the classifiers, we employ pattern matching tool to predict the label.

We believe that System 1 can perform better than the baseline because the classifiers only focus on one relation and do not need to consider other relations. In addition, it is easier to embed more feature for certain relation rather than the baseline.

System 2 used the same classifiers and pattern matching of System 1. The difference is that System 2 has post processing mechanism incorporated which checks and filters the output. Figure 6.3 illustrates of our System 2 approach.

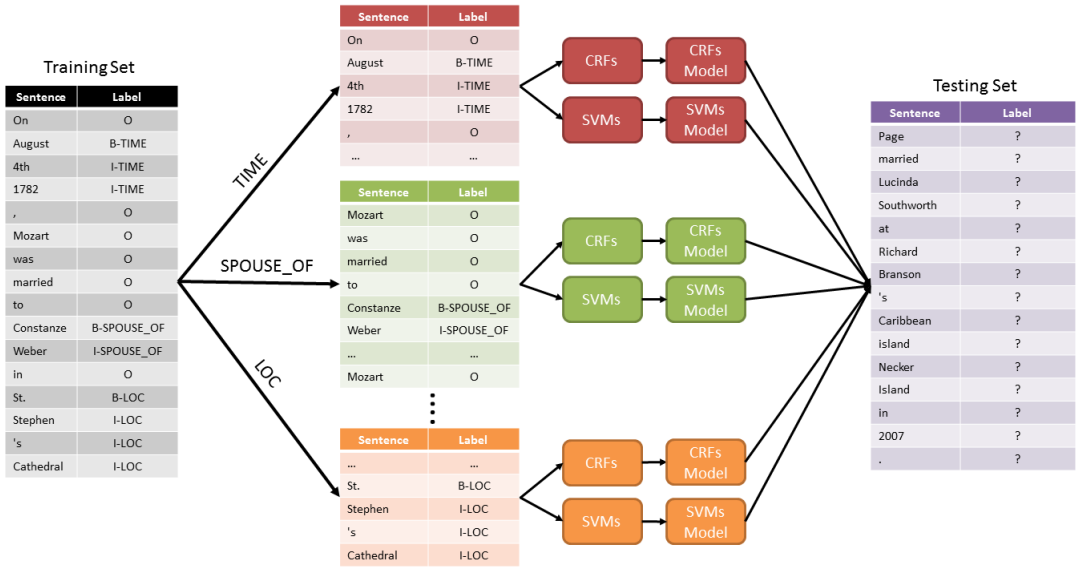

Figure 6.2: Classifiers for System 1

6.2 Experiment 2

In this experiment, we examined the connection between the size of training data compared to the classifiers performance. The amount of data that we used is gradually increased from 20%, 40%, 60% and 80%. Furthermore, we use the System 2 to check the performance for different training data size. However, we do not apply pattern matching in this experiment because it is independent of the size of the training data.

6.3 Evaluation

We use F-score (with B = 1) to measure the performance. The F-score is calculated as defined below

$$F\text{-}score = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{6.1}$$

Precision is the number of true positive over the sum of true positive and false positive, while the recall is the number of true positive over the sum of true positive and false negative. Since we are working with sequence classification task, the true positive is measured if the classifier predicts the full chunk perfectly. For example, if the chunk consists of three words and the classifiers only able to predict the correct label for the first two words, then the whole chunk is considered incorrect.

Table 6.1 illustrates the result of the classifiers.

In Table 6.1, the last two columns present the reference annotation and the classifiers'

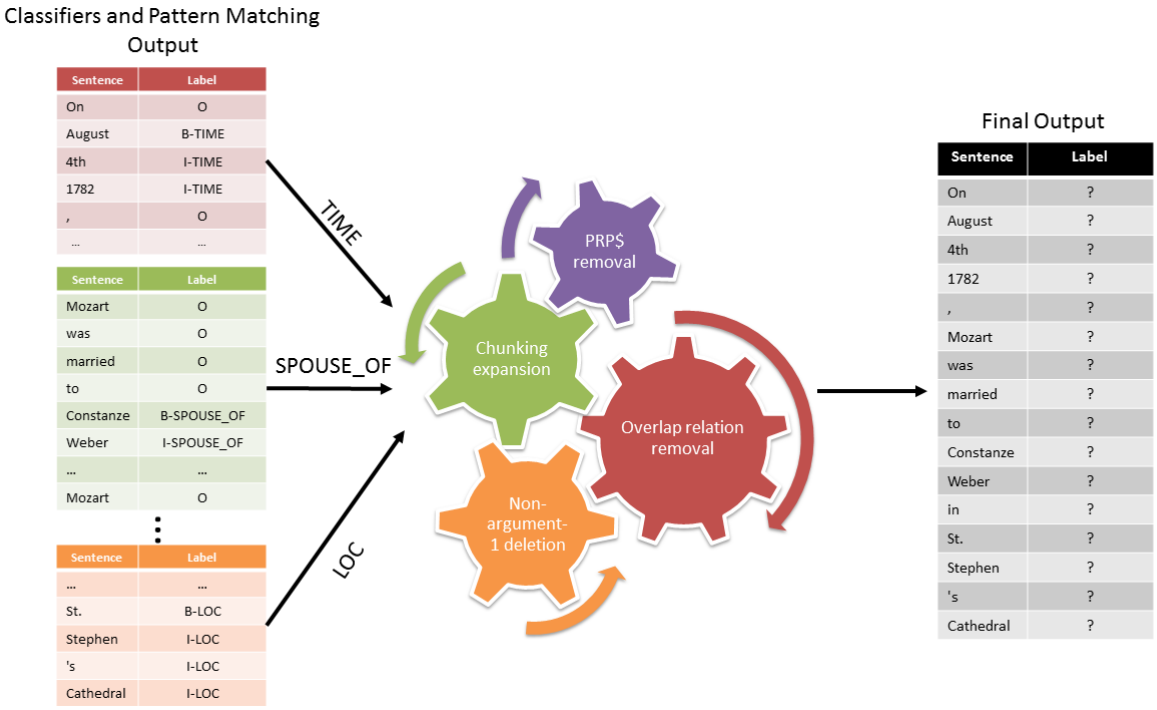

Figure 6.3: Post processing mechanism in System 2

SID	WID	Word	Lemma	POS	Stanford NER	Chunking	Illionis NER	Saarland NER	Key	Capital.	Reference	Prediction
1	0	he	he	PRP	O	B-NP	O	O	—	TRUE	O	O
1	1	is	be	VBZ	O	B-VP	O	O	—	FALSE	O	O
1	2	an	a	DT	O	B-NP	O	O	—	FALSE	O	O
1	3	american	american	JJ	MISC	I-NP	NORP-B	B-NORP:NATIONALITY	—	TRUE	O	O
1	4	business	business	NN	O	I-NP	O	B-JOB_TITLE	—	FALSE	B-TITLE	B-TITLE
1	5	magnate	magnate	NN	O	I-NP	O	I-JOB_TITLE	—	FALSE	I-TITLE	O
1	6	,	,	,	O	O	O	O	—	FALSE	O	O
1	7	investor	investor	NN	O	B-NP	O	B-JOB_TITLE	—	FALSE	B-TITLE	B-TITLE
1	8	,	,	,	O	O	O	O	—	FALSE	O	O
1	9	programmer	programmer	NN	O	B-NP	O	B-JOB_TITLE	—	FALSE	B-TITLE	B-TITLE
1	10	,	,	,	O	O	O	O	—	FALSE	O	O
1	11	inventor	inventor	NN	O	B-NP	O	B-JOB_TITLE	—	FALSE	B-TITLE	B-TITLE
1	12	and	and	CC	O	I-NP	O	O	—	FALSE	O	O
1	13	philanthropist	philanthropist	NN	O	I-NP	O	B-JOB_TITLE	—	FALSE	B-TITLE	O
1	14	.	.	.	O	O	O	O	—	FALSE	O	O

Table 6.1: Evaluation calculation

prediction. In the reference column, there are five TITLE labels (*business magnate*, *investor*, *programmer*, *inventor* and *philanthropist*) while the classifiers predict four TITLE labels where the first one is incorrect and the rest are correct). The precision and recall are calculated as:

$$Precision = \frac{3}{4}; Recall = \frac{3}{5} \tag{6.2}$$

$$F\text{-}score = 2 \cdot \frac{\frac{3}{4} \cdot \frac{3}{5}}{\frac{3}{4} + \frac{3}{5}} = 0.667 \tag{6.3}$$

To calculate this, we employ the tool that is developed to evaluate CoNLL-2000 shared task [Tjong Kim Sang and Buchholz, 2000].

Chapter 7

Results and Analysis

This chapter describes the experimental outcome and detailed evaluation. First set of the experiments is explained in Section 7.1. Section 7.1.1 then discusses the performance of our baseline system. In Section 7.1.2, the results from our System 1 where the classifiers are trained separately for each relation, are explained. The results of integrating post processing mechanism into our system is detailed in Section 7.1.3. Finally, Section 7.2 describes the learnability of the classifiers.

7.1 Experiment 1 results

In this experiment, 5-fold cross validation was applied to three systems which are the baseline, System 1 and System 2. The results from each system are explained in the next three sections below.

7.1.1 Baseline

Table 7.1 provides the result for the baseline. The first column describes the list of 29 semantic relations, which are our focus at the moment. The second column shows the best classifier tool for the particular relation. In the next three columns, the average precision, average recall and average F-score from 5-fold cross validation test are shown.

Most of the time, CRF++ shows a better performance for 12 relations followed by CRFSuite with LBFGS, and SVM^{hmm}. The best result is achieved for AGE_OF relation (with 85% F-score), followed by TIME (84.56% F-score) and TIME_BIRTH (78.7% F-score). By contrast, we had a hard time working with SUPPORTER_OF, PART_IN and FOUNDER_OF relation. The main reason is that it is hard to predict SUPPORTER_OF and FOUNDER_OF

Relation Name	Tools	AVG P	AVG R	AVG F-Score
ACCOMPLISHMENT	CRF++	68.70	43.80	53.19
AGE_OF	CRFSuite_LBFGS	93.96	78.01	85.02
AWARD	CRFSuite_LBFGS	69.97	48.69	56.70
CHILD_OF	CRFSuite_LBFGS	49.70	58.34	53.60
COLLEAGUE_OF	CRF++	52.40	31.79	38.58
CREATOR_OF	CRFSuite_LBFGS	40.72	19.86	25.56
DURATION	CRF++	91.33	60.81	72.50
EDUCATION_OF	CRFSuite_LBFGS	60.40	68.44	63.36
EMPLOYEE_OF	CRFSuite_LBFGS	34.98	14.29	18.65
FOUNDER_OF	SVM^{hmm}	20.21	15.46	16.95
LOC	CRF++	49.21	51.99	50.18
LOC_BIRTH	CRF++	62.40	68.00	64.68
LOC_DEATH	SVM^{hmm}	55.00	25.79	34.55
LOC_RESIDENCE	CRF++	61.29	41.89	49.00
MEMBER_OF	CRF++	50.79	28.53	35.20
NATIONALITY	CRF++	84.00	68.51	74.17
OWNER_OF	SVM^{hmm}	52.18	49.63	48.97
PARENT_OF	SVM^{hmm}	59.13	48.05	51.07
PART_IN	CRF++	22.44	8.54	11.43
RELIGION	SVM^{hmm}	36.00	23.33	27.36
SIBLING_OF	SVM^{hmm}	59.04	42.95	49.05
SPOUSE_OF	CRF++	71.67	27.61	38.82
SUBORDINATE_OF	CRFSuite_LBFGS	50.00	11.34	18.23
SUPPORTEE_OF	SVM^{hmm}	53.91	34.79	41.50
SUPPORTER_OF	-	0.00	0.00	0.00
TIME	CRF++	81.71	87.65	84.56
TIME_BIRTH	CRFSuite_LBFGS	86.21	73.11	78.74
TIME_DEATH	SVM^{hmm}	43.57	27.02	33.07
TITLE	CRF++	74.15	70.39	72.19

Table 7.1: Baseline result

relations due to the lack of instances. This problem is more complex because the classifiers are trained to learn all 29 relations at once, therefore the number of generated hypotheses can increase significantly and these affect the prediction quality. In addition, there is always trade off with other relations if we want to increase the quality of one relation.

For PART_IN relation, although the number of instances is larger than the other two, but the words that are used here is quite varied. In addition, it is quite often that the correct labels consist of proper name (e.g. iMac, iTunes, Encarta) which cannot be detected by the NER tagger.

7.1.2 System 1

The results for the second experiment are shown in Table 7.2. The format is similar to the previous table, we add one column to the right which is the baseline F-score. We acknowledged that pattern matching works really well for some relations, such as FOUNDER_OF, SUBORDINATE_OF, SIBLING_OF, PARENT_OF, and many more. From our investigation, we found that the patterns that we defined are representative enough to capture the necessary information.

In terms of the classifiers, the use of keyword as a feature also supports the performance for TIME_DEATH, LOC_DEATH, LOC_BIRTH and many more. Nevertheless, for TIME and LOC relations, the performance is lower compared to the baseline. This low performance is mainly caused by low recall or in other terms, the classifier often missed to predict the correct label.

In general, when the classifiers are trained for each relation separately, a better performance is achieved because it concentrates on one relation which does not affect the others relation. Furthermore, it gives us more flexibility in terms of feature selection of the classifiers. The additional feature which is the keyword that we embedded in this experiment, improves the performance significantly. Moreover, Saarland NER is good for detecting job title, religion and nationality in the text.

Apart from the positive impact, there are also some problems, which need to be resolved. First, we found that job title detection using Saarland NER could cause a problem. For example:

Example 15. *In 1901 Einstein became a Swiss* ***citizen****.*

In above example, Saarland NER mistakenly categorizes the word '*citizen*' as job title, which is considered as a good feature for the classifiers to label the word '*citizen*' as TITLE. To overcome this problem, we added a rule to filter the result that consists of this word in System 2.

Relation Name	Tools	AVG P	AVG R	AVG F-Score	Baseline	diff.
ACCOMPLISHMENT	CRF++	72.87	44.38	55.13	53.19	1.94
AGE_OF	SVM^{hmm}	95.11	77.77	85.39	85.02	0.36
AWARD	CRF++	80.37	62.33	69.48	56.70	12.77
CHILD_OF	pattern	85.92	87.23	86.55	53.60	32.95
COLLEAGUE_OF	CRFSuite_AP	51.92	39.78	39.94	38.58	1.36
CREATOR_OF	CRFSuite_AP	45.13	25.46	32.28	25.56	6.73
DURATION	pattern	88.40	68.30	76.52	72.50	4.02
EDUCATION_OF	CRF++	83.54	64.98	72.18	63.36	8.82
EMPLOYEE_OF	CRFSuite_AP	56.85	27.29	35.19	18.65	16.54
FOUNDER_OF	pattern	72.19	70.55	70.93	16.95	53.98
LOC	CRF++	60.83	33.79	42.81	50.18	-7.36
LOC_BIRTH	CRF++	94.15	83.70	88.52	64.68	23.84
LOC_DEATH	CRF++	90.00	54.96	67.16	34.55	32.62
LOC_RESIDENCE	CRFSuite_lbfgs	84.84	56.17	67.14	49.00	18.13
MEMBER_OF	pattern	36.04	46.67	39.08	35.20	3.88
NATIONALITY	CRF++	91.54	73.04	80.64	74.17	6.46
OWNER_OF	SVM^{hmm}	63.41	45.59	50.96	48.97	1.99
PARENT_OF	pattern	84.76	77.98	81.09	51.07	30.03
PART_IN	CRFSuite_AP	34.05	9.10	11.93	11.43	0.50
RELIGION	CRFSuite_AP	36.00	23.33	28.31	27.36	0.94
SIBLING_OF	pattern	92.89	85.03	88.13	49.05	39.08
SPOUSE_OF	pattern	78.54	62.63	69.51	38.82	30.69
SUBORDINATE_OF	CRFSuite_AP	63.07	34.99	43.10	18.23	24.86
SUPPORTEE_OF	SVM^{hmm}	83.33	46.97	58.00	41.50	16.50
SUPPORTER_OF	CRFSuite_AP	20.00	6.67	9.52	0.00	9.52
TIME	CRF++	85.54	83.53	84.50	84.56	-0.06
TIME_BIRTH	CRF++	89.52	89.40	89.03	78.74	10.29
TIME_DEATH	SVM^{hmm}	85.98	87.39	86.49	33.07	53.42
TITLE	CRF++	81.55	66.72	73.38	72.19	1.19

Table 7.2: System 1 result

Another problem occurs because the system does not have any preferences if multiple labels exist for the same chunk. For example:

Example 16. *Taylor died of congestive heart failure in* ***March 2011*** *at the age of 79, having suffered many years of ill health.*

The chunk '*March 2011*' is labeled as TIME and TIME_DEATH. Within this experiment, there is no preference/priority to select either TIME or TIME_DEATH for the final result.

Moreover, for TITLE relation, usually the chunk boundaries are not perfect especially if there is a prepositional phrase (PP) after the NP. For example:

Example 17. *She later became the president of property developer SC asset and* ***managing director*** *of Advanced Info Service.*

From the above example, we want to capture '*managing director of Advanced Info Services*' as TITLE, but the classifiers only marked '*managing director*' as TITLE due to the end of the NP and the PP information only covers the preposition. Therefore, we add chunk expansion to System 2 to get the full boundaries of the chunk.

From those examples, we concluded that using pattern matching and training the classifiers separately for each relation are not enough. We need some post processing mechanism to be applied in order to filter and finalize the result.

Apart from features and labels, there is a concern if we deal with a large number of relations in the future. To manually select the classifiers' feature or create a pattern matching for each relation could take a lot of time and efforts.

7.1.3 System 2

The problems that we defined in the previous section are handled by System 2. Basically we can say that System 2 is a refinement of System 1 because we added post processing pipeline to filter the output. The result for System 2 can be found in Table 7.3. The post processing used priority checking in selecting the final label for the chunk that are assigned to more than one label. This checking is done by seeing the results that is achieved in the previous system. However, only several relations use this priority checking.

Table 7.3 shows the results for this experiment. From this table, we observe the improvement (up to 4.11% compared to the previous experiment) for relations such as COLLEAGUE_OF, SUPPORTEE_OF, LOC, RELIGION and many more. The highest F-score is achieved for TIME_BIRTH with 90.42% followed by LOC_BIRTH with 88.52% and SIBLING_OF with 88.13%. The lowest F-score is obtained when classifying PART_IN and

Relation Name	Tools	AVG P	AVG R	AVG F-Score	System 1	diff.
ACCOMPLISHMENT	CRF++	72.87	44.38	55.13	55.13	0.00
AGE_OF	CRFSuite_AP	89.93	84.33	86.89	85.39	1.50
AWARD	CRF++	80.37	62.33	69.48	69.48	0.00
CHILD_OF	pattern	87.14	87.23	87.13	86.55	0.58
COLLEAGUE_OF	CRFSuite_AP	59.96	39.78	44.05	39.94	4.11
CREATOR_OF	CRFSuite_AP	47.18	25.15	32.58	32.28	0.29
DURATION	pattern	89.45	68.30	76.99	76.52	0.47
EDUCATION_OF	CRF++	84.11	64.98	72.39	72.18	0.21
EMPLOYEE_OF	CRFSuite_AP	56.85	27.29	35.19	35.19	0.00
FOUNDER_OF	pattern	74.10	70.55	71.81	70.93	0.88
LOC	CRFSuite_AP	68.52	35.54	46.39	42.81	3.58
LOC_BIRTH	CRF++	94.15	83.70	88.52	88.52	0.00
LOC_DEATH	CRF++	90.00	54.96	67.16	67.16	0.00
LOC_RESIDENCE	CRFSuite_lbfgs	87.36	55.46	67.23	67.14	0.10
MEMBER_OF	pattern	45.85	42.62	42.19	39.08	3.12
NATIONALITY	CRF++	92.35	73.04	81.03	80.64	0.40
OWNER_OF	SVM^{hmm}	63.41	45.59	50.96	50.96	0.00
PARENT_OF	pattern	85.69	77.98	81.49	81.09	0.40
PART_IN	CRFSuite_AP	34.76	9.10	12.14	11.93	0.21
RELIGION	SVM^{hmm}	50.00	31.67	31.33	28.31	3.03
SIBLING_OF	pattern	92.89	85.03	88.13	88.13	0.00
SPOUSE_OF	pattern	78.54	62.63	69.51	69.51	0.00
SUBORDINATE_OF	pattern	72.14	60.56	64.80	43.10	21.70
SUPPORTEE_OF	SVM^{hmm}	86.00	46.97	59.21	58.00	1.21
SUPPORTER_OF	CRFSuite_AP	23.33	6.67	10.13	9.52	0.61
TIME	CRFSuite_lbfgs	90.44	82.24	86.13	84.50	1.63
TIME_BIRTH	CRF++	92.37	89.40	90.43	89.03	1.39
TIME_DEATH	SVM^{hmm}	85.98	87.39	86.49	86.49	0.00
TITLE	CRF++	84.20	65.65	73.74	73.38	0.37

Table 7.3: System 2 result

SUPPORTER_OF relations. We believe that insufficient number of training instances and difficulty to find the boundaries of the chunk cause the low F-score.

In general, the idea of this post processing is to improve the prediction quality, which is done by reducing the false negative. For example:

Example 18. *In 1877 he went to Berlin for a year of study with physicists* ***Hermann von Helmholtz*** *and* ***Gustav Kirchhoff*** *and mathematician* ***Karl Weierstrass****.*

Our classifiers mark the chunks in Example 18 with SUBORDINATE_OF and COLLEAGUE_OF label. Since the F-score of SUBORDINATE_OF is higher than COLLEAGUE_OF, we omit the COLLEAGUE_OF from the chunk. Most of the problems that we mentioned earlier in subsection 4.6 are handled carefully. Nevertheless, some problems remain such as multiple labels among certain relations. For example:

Example 19. *He served as* ***the commander-in-chief of the Continental Army*** *during the American Revolutionary War.*

The chunk '*the commander-in-chief of the Continental Army*' is marked as TITLE, while the chunk '*Continental Army*' is marked as MEMBER_OF. From our perspective, it is valid to say that the person who leads the army also considered as a member of the army. Another interesting example:

Example 20. *Living in Johannesburg, he became involved in anti-colonial politics, joining the ANC and becoming a founding member of its* ***Youth League****.*

In above example, the chunk '*Youth League*' got FOUNDER_OF and MEMBER_OF label. Again, from our point of view, both of these labels are correct since the person founded the organization and became a member of this organization. Example 19 and 20 briefly give us important information that it is necessary to put additional label to the chunks rather than keep it as single label.

7.2 Experiment 2 results

Figure 7.1 provides the learning curve for all 29 relations that we have. From this figure, we acknowledged that the larger training data positively correlates with the higher F-score. SUPPORTEE_OF relation is the most sensitive the amount of training data followed by LOC_DEATH and SUBORDINATE_OF.

Figure 7.2 shows the final output from our AEM. The green mark represents the chunk label that is predicted by our AEM and matched with the reference annotation. The red

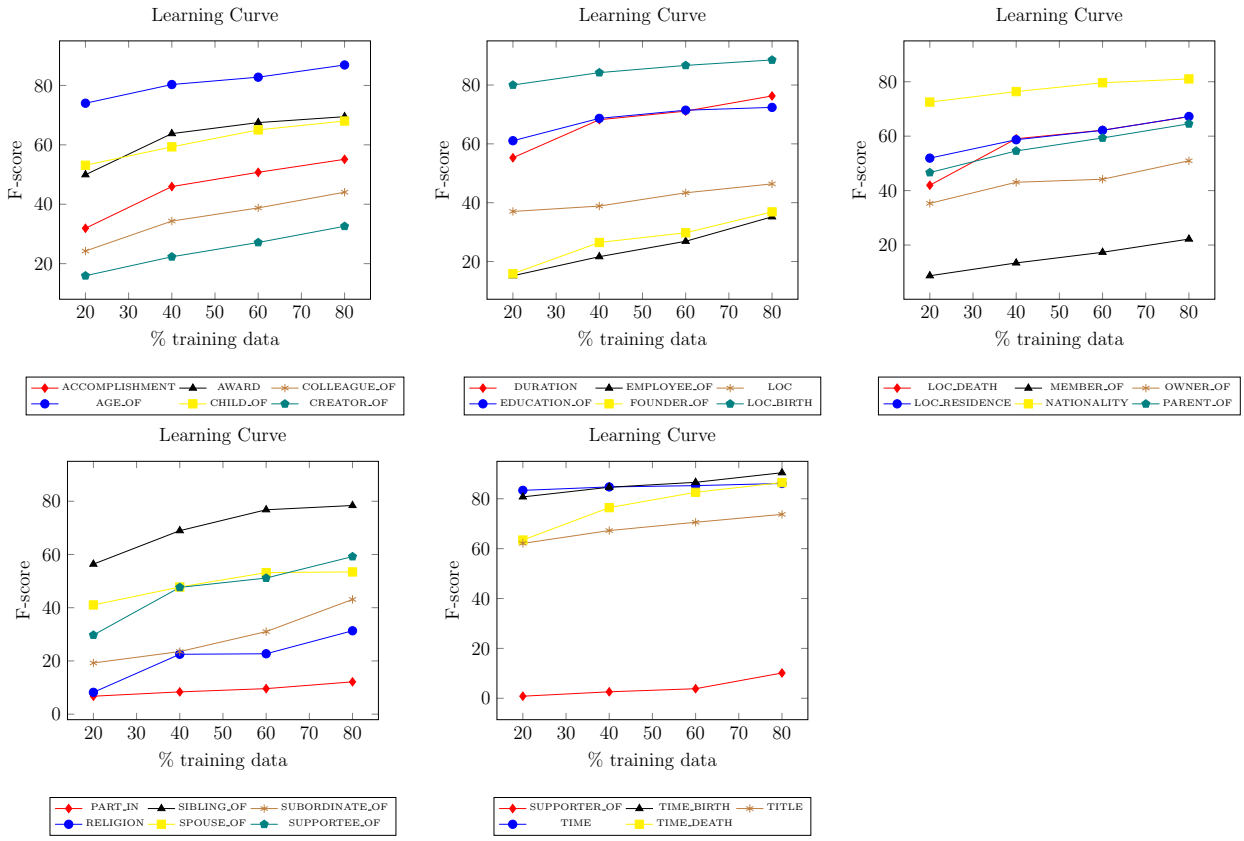

Figure 7.1: Learning curves for the defined relations

mark represents the chunk label that is missed by our AEM. In addition, the yellow mark shows that there are more than one output from our AEM for the same chunk and the orange mark describes the chunk that is predicted incorrectly .

Figure 7.2: AEM final output

Chapter 8

Conclusions

In this report, we have presented our AEM to extract biographical facts of famous people automatically from unstructured data. We started this work by collecting the data, which are raw texts from Wikipedia. After that, some preprocessing mechanisms were applied to clean the data. In addition, we defined a number of semantic relations in order to portray the relevant facts.

To extract these relations automatically, we developed a system that consists of pattern matching and sequence classifiers such as CRFs and SVMs. Features such as word and lemma tokens, POS tags, chunk information, NE tags, capitalization and keyword were employed to train classifiers. Moreover, post processing steps such as possessive pronoun removal, non-argument-1 removal, relation overlap removal and chunking expansion were utilized to filter and finalize the output.

we conducted two sets of experiments in this work. In the first experiment we designed three systems, which are the Baseline, System 1, and System 2 to learn and predict defined relations. The baseline consists of the classifiers that learn and predict output for all relations in one step. In System 1, the classifiers were trained separately for each relation. Additionally, pattern matching was utilized to see whether it can increase the prediction quality or not. System 2 had the same classifiers and pattern matching tool as System 1, except for additional post processing to check and finalize the results.

From the first experiment, we observed that System 1, in general, outperformed the baseline by a large margin. The flexibility to select the features for each relation's classifiers, additional keyword feature and pattern matching were the reasons why the F-score for System 1 was higher than the baseline. With additional post processing steps in System 2, the prediction quality can be improved up to 21% in terms of F-score. Furthermore, our system performed well (F-score $\geq 80\%$) for some relations such as TIME_BIRTH, SIBLING_OF, CHILD_OF, LOC_BIRTH, TIME_DEATH, TIME , PARENT_OF and NATIONALITY.

However, the prediction quality for PART_IN, RELIGION, EMPLOYEE_OF and CREATOR_OF relations need to be improved.

Another experiment that we run is to obtain the learning curve of the model given different size of training set. Our results showed that larger training data positively correlates with a higher F-score. SUPPORTEE_OF, LOC_DEATH, SUBORDINATE_OF were the most sensitive to the amount of training data.

8.1 Future Work

There is a lot of interesting ideas that can be pursued and developed in further research. From our observation, when the focus lies on only one relation, some of the instances were correctly predicted by the classifiers and not with pattern matching (and vice versa). Therefore, it will be interesting to see whether combining the results from the classifiers and pattern matching can improve the overall performance rather, than just selecting the best outcome from both.

Looking back to the systems' performance, we realized that the keyword had played an important role so that the classifiers can predict the correct label. Right now, our keyword is not detected automatically. We selected the keyword for a certain relation ad hoc based on intuition. In addition, to select the manual keyword for larger set of relations is time and cost consuming. Thus, it is interesting to see how the keyword can be expanded using semantic information from semantic databases such as WordNet[1]. Alternatively, we can use automatic event detection to find the sentence focus.

A similar approach can be applied for the manual pattern matching where regular expressions that we set are tailored to handle this particular data. However, to get more coverage using this, maybe we can use the work that has been done by Ravichandran and Hovy [2002]. The idea is to use bootstrapping to create automatic pattern from the search engine. Some QA pairs examples are fed into the search engine and then the system retrieves the result to make an automatic pattern based on the words between the question query and the answer.

From the evaluation point of view, the results that we have at the moment are evaluated by using F-score which is very strict. If there is one word that is missing from the whole chunk, then it will be judged as incorrect prediction. If we see Example 18, in our reference annotation, we marked '*with physicists Hermann von Helmholtz*', '*Gustav Kirchhoff*', and '*mathematician Karl Weierstrass*' with SUBORDINATE_OF. However, the system only found the name of the person such as '*Hermann von Helmholtz*' and '*Karl Weierstrass*'

[1]http://wordnet.princeton.edu

(without the title mathematician and physicist). These output are considered incorrect due to the fact that the prediction does not fit with the one in the reference annotation. However, as humans, we can say these predictions are correct. In fact, the focus of the chunk, which is the name of the person, has already been identified. Therefore, we need a more lenient evaluation metric so the results of our AEM can be measured better.

From the semantic relations point of view, we are aware that one chunk may be assigned to more than one relation (See Example 19 and 20). Therefore, in the future research, we should accommodate these multiple relations labels in our reference annotation data. Besides that, some relations definition may need to be sharpened, especially the relations that cannot be detected by NER tagger. In addition, providing more training examples for these relations will be helpful since we have seen that more training data would increase the system performance as well.

Last, it might be useful if in the future we can also integrate knowledge based approaches such as Freebase[2] and Ontologies. Although the quality in Freebase is hard to evaluate, but with the crowdsourcing, more important and relevant facts can be added to the database. This knowledge based can support our AEM in finding the answer that may not exist from the raw text. An extension to our AEM from Freebase is described in Appendix B.

[2]http://www.freebase.com/

Appendices

Appendix A

Relations definition

In this chapter we provide 17 relations definition that we mapped from TAC KBP Slot Filling task

1. AGE_OF(Z,X)
   Content: Value or string
   Quantity: Single
   Description: A reported age of the assigned person.

   (a) Age of the person at death is an acceptable answer.

   (b) Previous ages are valid responses.

   (c) Approximate ages are valid responses. For example, if a source document states that the assigned person was about 50, then 50 would be a valid filler.

Entity	**Document passage**	**Correct Filler**
Franz Josef	Tiger Woods was introduced to golf before the age of two by his athletic father Earl.	Two
Adam Smith	Smith entered the University of Glasgow when he was fourteen and studied moral philosophy under Francis Hutcheson.	Fourteen
Johann Sebastian Bach	on 28 July 1750 Bach died at the age of 65.	65

Table A.1: AGE_OF examples

2. CHILD_OF(Z,X)
   Content: Name
   Quantity: List
   Description: The parents of the assigned person. In addition to biological parents, step-parents and adoptive parents are also acceptable answers.

Entity	Document passage	Correct Filler
Franz Josef	He was the oldest son of Archduke Franz Karl and his wife Princess Sophie of Bavaria.	Archduke Franz Karl, Princess Sophie of Bavaria
Alexander the Great	He was the son of the king of Macedon , Philip II, and his fourth wife, Olympias .	The king of Macedon Phillip II Olympias
Marco Polo	Marco polo learned the mercantile trade from his father and uncle, Niccolo and Maffeo .	niccolo

Table A.2: CHILD_OF examples

3. EDUCATION_OF(Z,X)
   Content: Name
   Quantity: List
   Description: Any school (college, high school, university, etc.) name that the assigned person has attended.

   (a) Enrollment at a school does justify a filler for this slot as well.

Entity	Document passage	Correct Filler
Bill Gates	Gates graduated from Lakeside School in 1973.	Lakeside School
George W. Bush	Bush attended Yale University from 1964 to 1968, graduating with an B.A. in history.	Yale University
Johannes Kepler	in 1589, after moving through grammar school, latin school, and seminary at Maulbronn, Kepler attended Tubinger Stift at the University of Tubingen.	University of Tubingen

Table A.3: EDUCATION_OF examples

4. EMPLOYEE_OF(Z,X)
   Content: Name
   Quantity: List
   Description: The organizations or geopolitical entities (governments) by which the assigned person has been an employee.

   (a) Actors, directors, screenwriters, etc. should not be considered employees of TV shows or movies in which they appeared or helped to produce. However, they should be considered as employees of networks/companies that produced such works.

   (b) Organizations of which the assigned entity is a founder or owner with no other position/relationship are not acceptable answers.

   (c) If in the sentence contains the words such as, society, congress, group, party, member, fellow then it is considered to be part of MEMBER_OF*member_of*.

5. LOC_BIRTH(Z,X)
   Content: Name

Entity	Document passage	Correct Filler
Steve Jobs	After a power struggle with the board of directors in 1985, Jobs left Apple and founded Next.	Apple
Michael Bloomberg	Bloomberg began his career at the securities brokerage Salomon Brothers.	the securities brokerage Salomon Brothers
Rntgen	in 1875 he became a professor at the Academy of Agriculture at Hohenheim, Wrttemberg.	the Academy of Agriculture, Hohenheim

Table A.4: EMPLOYEE_OF examples

Quantity: Single

Description: The place (village, city, town, village, province, state, country, etc.) where the assigned person was born.

Entity	Document passage	Correct Filler
Lady Gaga	She was born and raised in New York City.	New York City
Mehmet Oz	Oz was born in Cleveland, Ohio, to Suna and Mustafa z.	Cleveland, Ohio
Martin Luther King, Jr	Martin Luther King, Jr. was born on January 15, 1929, in Atlanta, Georgia.	Atlanta, Georgia

Table A.5: LOC_BIRTH examples

6. LOC_DEATH(Z,X)

   Content: Name

   Quantity: Single

   Description: The place (village, city, town, village, province, state, country, etc.) where the assigned person died.

Entity	Document passage	Correct Filler
Franz Joseph	Franz Joseph died in the Schoenbrunn palace in 1916.	Schoenbrunn palace
Soekarno	He died of kidney failure in Jakarta Army Hospital on 21 june 1970 at age 69.	Jakarta Army Hospital
Marie Antoinette	she was beheaded at the Place de la Revolution.	Place de la Revolution

Table A.6: LOC_DEATH examples

7. LOC_RESIDENCE(Z,X)

   Content: Name

   Quantity: List

   Description: The place (village, city, town, village, province, state, country, etc.) where the assigned person has lived/did live/is living.

   (a) Former places of residence are correct fillers

8. MEMBER_OF(Z,X)

   Content: Name

Entity	Document passage	Correct Filler
Madonna	She moved to New York City.	New York City
Louis Pasteur	Louis grew up in the town of Arbois.	The town of Arbois
Johannes Gutenberg	until at least 1444 he lived in Strasbourg, most likely in the St. Arbogast parish.	Strasbourg, St. Arbogast parish

Table A.7: LOC_RESIDENCE examples

Quantity: List

Description: The organizations or geopolitical entities (governments) by which the assigned person has been a member.

(a) The words such as, society, congress, group, party, member, fellow can be used as justification for membership.

Entity	Document passage	Correct Filler
Max Planck	in berlin , planck joined the local Physical Society.	The local Physical Society
Jeremy Lin	Lin joined the Dallas Mavericks for mini-camp as well as their NBA summer league team in Las Vegas.	Dallas Mavericks, NBA summer league team
Julia Gillard	Gillard became the first female Deputy Prime Minister of Australia upon Labor 's victory in the 2007 federal election.	Labor

Table A.8: MEMBER_OF examples

9. NATIONALITY(Z,X)

Content: Name

Quantity: List

Description: The nationality/origin and/or ethnicity of the assigned person.

(a) Former nationalities are acceptable responses.

(b) When both nationality and ethnicity are mentioned in source documents, both are valid answers. For example, if a document states that the assigned person is a Chinese American, both Chinese and American would be correct fillers.

(c) If country of birth is stated, nationality can be inferred unless specifically stated otherwise (such as "Egyptian-born Canadian").

Entity	Document passage	Correct Filler
Steve Jobs	He was an American entrepreneur and inventor.	American
Oscar Wilde	He was an Irish writer and poet .	Irish
Saladin	A Muslim of Kurdish origin , Saladin led Islamic opposition against the European Crusaders in the Levant.	Kurdish

Table A.9: NATIONALITY examples

10. PARENT_OF(Z,X)

Content: Name or string

Quantity: List

Description: The children of the assigned person, including adopted and step-children.

Entity	Document passage	Correct Filler
Albert Einstein	They had two sons	Two sons
Angelina Jolie	On March 10 , 2002, Jolie adopted her first child , seven-month-old Maddox Chivan, from an orphanage in Phnom Penh, Cambodia	First child, Maddox Chivan
Vladimir Putin	Putin and his wife have two daughters, Mariya Putina and Yekaterina Putina .	Two daughters, Mariya Putina, Yekaterina Putina

Table A.10: PARENT_OF examples

11. RELIGION(Z,X)

Content: String

Quantity: Single

Description: The religion to which the assigned person has belonged. Former (but not future or potential) religions are acceptable answers.

Entity	Document passage	Correct Filler
Russell Crowe	He was baptized as a Christian	Christian
Malala Yousafzai	Malala Yousafzai was born into a Muslim family of Pashtun ethnicity in July 1997 and given her first name, Malala , meaning 'grief stricken'.	Muslim
John Nash	They married in February 1957 at a Catholic ceremony , although Nash was an Atheist.	Atheist

Table A.11: RELIGION examples

12. SIBLING_OF(Z,X)

Content: Name or string

Quantity: List

Description: The religion to which the assigned person has belonged. Former (but not future or potential) religions are acceptable answers.

(a) In addition to full siblings, step-siblings and half-siblings are acceptable answers.

(b) Brother(s) or sister(s)-in-law are not acceptable responses for siblings (they are fillers for OTHER_FAMILY).

Entity	Document passage	Correct Filler
Michael Jackson	He debuted on the professional music scene along with his brothers as a member of The Jackson 5 in 1964.	brothers
Mark Zuckerberg	he and his three sisters, Randi, Donna, and Arielle, were brought up in Dobbs Ferry, New York.	Three sisters, Randi, Donna, Arielle
John Dalton	He joined his older brother Jonathan at age 15 in running a Quaker school in nearby Kendal.	Older brother Jonathan

Table A.12: SIBLING_OF examples

13. SPOUSE_OF(Z,X)
    Content: Name
    Quantity: List
    Description: The spouse(s) of the assigned person.

    (a) Former spouses are acceptable answers.

    (b) Marriages do not have to be legally recognized in order for resulting spouses to be correct fillers. The word 'partner' also does justify a spouse filler, 'mistress' as well.

Entity	Document passage	Correct Filler
Franz Josef	On 24 April 1854 he married Elisabeth Amalie Eugene von Wittelsbach (aka Sisi)	Elisabeth Amalie Eugene von Wittelsbach Sisi
Galileo Galilei	Galileo fathered three children out of wedlock with Marina Gamba.	Marina Gamba
Diana	Diana , Princess of Wales was the first wife of Charles , Prince of Wales.	Charles

Table A.13: SPOUSE_OF examples

14. TIME_BIRTH(Z,X)
    Content: Value
    Quantity: Single
    Description: time when the assigned person was born.

Entity	Document passage	Correct Filler
Lady Gaga	Lady Gaga was born in 1986.	1986
Michael Jordan	Michael Jordan was born on February 17 , 1963 in Brooklyn , New York .	February 17 , 1963
Charles Darwin	Charles Robert Darwin (born 12 February 1809) was an English naturalist .	12 February 1809

Table A.14: TIME_BIRTH examples

15. TIME_DEATH(Z,X)
    Content: Value

Quantity: Single

Description: time when the assigned person died.

Entity	Document passage	Correct Filler
Albert Einstein	He died on April 18th 1955.	April 18th 1955
Steve Jobs	he died of respiratory arrest related to his metastatic tumor on October 5, 2011.	October 5, 2011
Martin Luther King Jr.	King was assassinated on April 4, 1968, in Memphis, Tennessee.	April 4, 1968

Table A.15: TIME_BIRTH examples

16. TITLE(Z,X)

Content: String

Quantity: List

Description: Official or unofficial name(s) of the employment and occupation/job positions and degrees within organizations have been held by the assigned person

Entity	Document passage	Correct Filler
Helen Keller	Helen Adams Keller was an american author, political activist, and lecturer.	Author, political activist, lecturer
Julius Caesar	Gaius Julius Caesar was a roman general, statesman, consul and notable author of latin prose.	General, statesman, consul, notable author
Aristotle	Aristotle was a greek philosopher and polymath.	Philosopher, polymath

Table A.16: TITLE examples

Appendix B

Freebase

Freebase is a huge open knowledge graph. It contains more than 20 million entities (person, place, thing, etc.) and it is open to the public for adding and modifying the information. In freebase the entities are connected in a form of graph. Like in the normal graph, knowledge graph also contains nodes and edges.

The node, which is called as a topic or entity, represents a single concept or real world thing. A topic in Freebase has a unique identifier, which is called machine id (MID). For example, MID /m/04lg6 represents an entity about Leonardo da Vinci.

The edge defines the relationship between one topic to another. A type in Freebase is introduced as an IS A relationship about a topic. For example, the topic Leonardo da Vinci is defined as a person, a visual artist, an architect, etc. In addition, a type can contain several properties. Each property is defined as a HAS A relationship between the topic and the property's value. This property's value can be a single or compound value type. It is considered compound value type if it consists of multiple fields. For example, Leonardo da Vinci is a visual artist who has artworks such as Mona Lisa, the Last Supper etc. Each of these artworks is considered as a single value. An example of compound value type is given in an example below:


```
"/people/person/sibling_s": {
 "valuetype": "compound",
 "values": [
  {
   "text": "Bartolomeo da Vinci - racqztorres24 - Sibling Relationship",
   "lang": "en",
   "id": "/m/0nf375x",
   "creator": "/user/racqztorres24",
   "timestamp": "2012-11-23T20:36:20.000Z",
```

```
    "property": {
     "/people/sibling_relationship/sibling": {
      "valuetype": "object",
      "values": [
       {
        "text": "Bartolomeo da Vinci",
        "lang": "en",
        "id": "/m/0nf375y",
        "creator": "/user/racqztorres24",
        "timestamp": "2012-11-23T20:36:20.000Z"
       }
      ],
      "count": 2.0
      ....
     }
    }
   },
   {
    "text": "Antonio Ser Piero - jaysondg - Sibling Relationship",
    "lang": "en",
    "id": "/m/0t_6pvt",
    "creator": "/user/jaysondg",
    "timestamp": "2013-05-05T19:22:45.002Z",
    "property": {
     "/people/sibling_relationship/sibling": {
      "val2.0
      ....
```


In above example, the 'valuetype' is defined as compound and there are multiple fields inside of this *sibling_s* property. From it, we know that Leonardo Da Vinci has two brothers *Bartolomeo da Vinci* and *Antonio Ser Piero*.

For the purpose of our work, we use types and properties of famous person and map it into our relation label. If we want to know the artworks that are created by Leonardo da Vinci, we can query this by typing:


```
[{
  "type": "/people/person",
  "id": "/m/04lg6",
  "/visual_art/visual_artist/artworks": []
}]
```

The output from Freebase is represented as below:

```
{
  "result": [{
    "/visual_art/visual_artist/artworks": [
      "Mona Lisa",
      "Ginevra de' Benci",
      "The Last Supper",
     ...
      "Madonna and Child with St Joseph",
      "Sala delle Asse",
      "Leonardo's horse"
    ],
    "id": "/m/04lg6",
    "type": "/people/person"
  }]
}
```

The above output is formatted using JSON (JavaScript Object Notation). A JSON object is considered as a dictionary or associative array which map pointer/key to value. This value can be literal (null, true, false), number, string, array or object.

B.1 Freebase Mapping

Table B.1 shows the mapping from Freebase types and properties to our semantic relations. The first column explains the type or property from Freebase. Since, one type may contains a compound value and this value can be another types and properties, we further include this information in the second column. Last column describes the corresponding semantic relation that matches with these types or properties.

Types or Properties (level 1)	Types or Properties (level 2)	Relation
/people/person/education	/education/education/degree	ACCOMPLISHMENT
/people/person/age		AGE_OF
/common/topic/alias		ALT_NAME
/award/award_nominee/award_nominations	/award/award_nomination/award	AWARD
/award/award_winner/awards_won	/award/award_honor/award	AWARD
/people/person/height_meters		BODY
/people/person/weight_kg		BODY
/people/person/parents		CHILD_OF
/influence/influence_node/peers	/influence/peer_relationship/peers	COLLEAGUE_OF
/book/author/works_written		CREATOR_OF
/music/lyricist/lyrics_written		CREATOR_OF
/music/artist/album		CREATOR_OF
/music/composer/compositions		CREATOR_OF
/law/inventor/inventions		CREATOR_OF
/chemistry/element_discoverer/discovered		CREATOR_OF
/visual_art/visual_artist/artworks		CREATOR_OF
/film/director/film		CREATOR_OF
/people/person/education	/education/education/institution	EDUCATION_OF
/music/artist/label		EMPLOYEE_OF
/people/person/employment_history	/business/employment_tenure/company	EMPLOYEE_OF
/celebrities/celebrity/celebrity_rivals	/celebrities/rivalry/rival	ENEMY_OF
/organization/organization_founder/organizations_founded		FOUNDER_OF
/celebrities/celebrity/celebrity_friends	/celebrities/friendship/friend	FRIEND_OF
/people/person/gender		GENDER
/people/person/place_of_birth		LOC_BIRTH
/people/deceased_person/place_of_death		LOC_DEATH
/people/person/places_lived	/people/place_lived/location	LOC_RESIDENCE
/organization/organization_member/member_of	/organization/organization_membership/organization	MEMBER_OF
/government/politician/party	/government/political_party_tenure/party	MEMBER_OF
/people/person/ethnicity		NATIONALITY
/people/person/nationality		NATIONALITY
/sports/sports_team_owner/teams_owned		OWNER_OF
/people/person/children		PARENT_OF
/film/person_or_entity_appearing_in_film/films	/film/personal_film_appearance/film	PART_IN
/film/actor/film	/film/performance/film	PART_IN
/tv/tv_personality/tv_regular_appearances	/tv/tv_regular_personal_appearance/program	PART_IN
/people/deceased_person/cause_of_death		REASON
/people/person/religion		RELIGION
/people/person/sibling_s	/people/sibling_relationship/sibling	SIBLING_OF
/people/person/spouse_s	/people/marriage/spouse	SPOUSE_OF
/education/academic/advisors		SUBORDINATE_OF
/government/political_appointer/appointees	/government/government_position_held/office_holder	SUBORDINATE_OF
/education/academic/advisees		SUPERIOR_OF
/business/employer/employees	/business/employment_tenure/person	SUPERIOR_OF
/people/person/date_of_birth		TIME_BIRTH
/people/deceased_person/date_of_death		TIME_DEATH
/people/person/profession		TITLE

Table B.1: Freebase mapping

Appendix C

List of Regular Expressions

For our pattern matching tool, we defined 149 regular expressions that are divided into 12 relations. The following tables list 11 relations and their regular expressions.

Regular Expression	Group No.	Relation
\s(work|worked|working|works) at (.*?) for\s	2	EMPLOYEE_OF
\s(work|worked|working|works) .* at (.*?) for\s	2	EMPLOYEE_OF
\s(work|worked|working|works) at (.*?) [\.,\/#!$%\^*;:{}=\-_~()]	2	EMPLOYEE_OF
\s(work|worked|working|works) .* at (.*?) [\.,\/#!$%\^*;:{}=\-_~()]	2	EMPLOYEE_OF
\s(work|worked|working|works) for (.*?) [\.,\/#!$%\^*;:{}=\-_~()]	2	EMPLOYEE_OF
\s(work|worked|working|works) .* for (.*?) [\.,\/#!$%\^*;:{}=\-_~()]	2	EMPLOYEE_OF
\scontract with (.*?) [\.,\/#!$%\^*;:{}=\-_~()]	1	EMPLOYEE_OF
\s(become|became) at (.*?) [\.,\/#!$%\^*;:{}=\-_~()]	2	EMPLOYEE_OF
\s(become|became) .* at (.*?) [\.,\/#!$%\^*;:{}=\-_~()]	2	EMPLOYEE_OF

Table C.1: EMPLOYEE_OF regular expressions

Regular Expression	Group No.	Relation
\s(died|death|assassinated|killed|murdered) on (.*) [\.,\/#!$%\^*;:{}=\-_~()]	2	TIME_DEATH
\s(died|death|assassinated|killed|murdered) .* on (.*) [\.,\/#!$%\^*;:{}=\-_~()]	2	TIME_DEATH
\s(died|death|assassinated|killed|murdered) in (.*) (and|[\.,\/#!$%\^*;:{}=\-_~()])	2	TIME_DEATH
\s(died|death|assassinated|killed|murdered) .* in (.*) [\.,\/#!$%\^*;:{}=\-_~()]	2	TIME_DEATH

Table C.2: TIME_DEATH regular expressions

Regular Expression	Group No.	Relation
\s(died|assassinated|beheaded) (in|at) (.*?) (on|[\.,\/#!$%\^*;:{}=\-_~()])	3	LOC_DEATH
\s(assassinated|assassination|died|beheaded|death) .* (in|at) (.*) [\.,\/#!$%\^*;:{}=\-_~()]	3	LOC_DEATH

Table C.3: LOC_DEATH regular expressions

Regular Expression	Group No.	Relation
\s(tutelage|assistant|student) of (.*) [\.,\/#!$%\^*;:{}=\-_~()]	2	SUBORDINATE_OF
\s(assistant|adviser|student|) to (.*) [\.,\/#!$%\^*;:{}=\-_~()]	2	SUBORDINATE_OF
\s(directed|mentored|recruited|tutored|taught|appointed) by (.*) [\.,\/#!$%\^*;:{}=\-_~()]	2	SUBORDINATE_OF
\s(directed|mentored|recruited|tutored|taught|appointed) .* by (.*) [\.,\/#!$%\^*;:{}=\-_~()]	2	SUBORDINATE_OF
\s(appointed|study with|under) (.*) [\.,\/#!$%\^*;:{}=\-_~()]	2	SUBORDINATE_OF
\swork for (.*) at\s	1	SUBORDINATE_OF
\slearn from (.*)	1	SUBORDINATE_OF
^(his|her) (supervisor|teacher) \W (.*) [\.,\/#!$%\^*;:{}=\-_~()]	3	SUBORDINATE_OF
^(his|her) (supervisor|teacher) (.*) [\.,\/#!$%\^*;:{}=\-_~()]	3	SUBORDINATE_OF
\s(his|her|'s) (supervisor|teacher|assistant|adviser|advisor) \W (.*) [\.,\/#!$%\^*;:{}=\-_~()]	3	SUBORDINATE_OF
\s(his|her|'s) (supervisor|teacher|assistant|adviser|advisor) (.*) [\.,\/#!$%\^*;:{}=\-_~()]	3	SUBORDINATE_OF

Table C.4: SUBORDINATE_OF regular expressions

Regular Expression	Group No.	Relation
^(she|he) married (.*)	2	SPOUSE_OF
^(his|her) (wife|husband|spouse) \W (.*)	3	SPOUSE_OF
^(his|her) \w* (wife|husband|spouse) \W (.*)	3	SPOUSE_OF
^(his|her) (wife|husband|spouse) (.*)	3	SPOUSE_OF
^(his|her) \w* (wife|husband|spouse) (.*)	3	SPOUSE_OF
\w* and (.*) were married	1	SPOUSE_OF
\w* and (.*) married	1	SPOUSE_OF
\w* and (.*) are parents	1	SPOUSE_OF
\w* married to (.*)	1	SPOUSE_OF
marriage was to (.*)	1	SPOUSE_OF
marriage to (.*)	1	SPOUSE_OF
married (his|her) (wife|husband|spouse) \W (.*)	3	SPOUSE_OF
married (his|her) (wife|husband|spouse) (.*)	3	SPOUSE_OF
\s(his|her) (wife|husband|spouse) \W (.*)	3	SPOUSE_OF
\s(his|her) \w* (wife|husband|spouse) \W (.*)	3	SPOUSE_OF
\s(his|her) (wife|husband|spouse) (.*)	3	SPOUSE_OF
\s(his|her) \w* (wife|husband|spouse) (.*)	3	SPOUSE_OF
\s(she|he) married (.*)	2	SPOUSE_OF
\w* married (.*)	1	SPOUSE_OF
(child|son|daughter|kid|boy|girl|baby|twin|children|sons|daughters|kids|boys|girls|babies|twins) with (.*)	2	SPOUSE_OF
\swedlock with (.*)	1	SPOUSE_OF
\swidow of (.*)	1	SPOUSE_OF
\s(first|second|third|fourth|fifth|sixth|seventh|eight|ninth|tenth) (wife|husband|spouse) of (.*)	3	SPOUSE_OF
\s(first|second|third|fourth|fifth|sixth|seventh|eight|ninth|tenth) (wife|husband|spouse) \W (.*)	3	SPOUSE_OF
\s(first|second|third|fourth|fifth|sixth|seventh|eight|ninth|tenth) (wife|husband|spouse) (.*)	3	SPOUSE_OF

Table C.5: SPOUSE_OF regular expressions

Regular Expression	Group No	Relation
\sage ([0-9]* (months|years){0,1})	1	AGE_OF
\saged ([0-9]* (months|years){0,1})	1	AGE_OF
\sage of about (\w*) .*	1	AGE_OF
\sage of (\S*) .*	1	AGE_OF
\sage (\S* (months|years){0,1})	1	AGE_OF
\saged (\S* (months|years){0,1})	1	AGE_OF
\sat age (.*) \W	1	AGE_OF
\s was ((.*) (months|years) old)	1	AGE_OF
\she was (\w*) \W	1	AGE_OF
\sshe was (\w*) \W	1	AGE_OF
\swas about (.*) \W	1	AGE_OF
\swas ([0-9]*) \W	1	AGE_OF
\sturned ([0-9]*)	1	AGE_OF
\swas around ([0-9]*)	1	AGE_OF
\sjust ([0-9]*)	1	AGE_OF
\shis (.*) birthday	1	AGE_OF
\sher (.*) birthday	1	AGE_OF

Table C.6: AGE_OF regular expressions

Regular Expression	Group No. 1	Group No. 2	Relation
^(her|his|with) (sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings) \W (.*)	3	2	SIBLING_OF
^(her|his|with) (sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings) (.*)	3	2	SIBLING_OF
^(her|his|with) (\w*\s*(sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings)) \W (.*)	4	2	SIBLING_OF
^(her|his|with) (\w*\s*(sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings)) (.*)	4	2	SIBLING_OF
\s(have|has|had) ((a|an) \w*\s*(sister|brother|sibling)) \W (.*)	5	2	SIBLING_OF
\s(have|has|had) ((a|an) \w*\s*(sister|brother|sibling)) called (.*)	5	2	SIBLING_OF
\s(have|has|had) ((a|an) \w*\s*(sister|brother|sibling)) (.*)	5	2	SIBLING_OF
\s(her|his|with) (sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings) \W (.*)	3	2	SIBLING_OF
\s(her|his|with) (sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings) (.*)	3	2	SIBLING_OF
\s(her|his|with) (\w*\s*(sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings)) \W (.*)	4	2	SIBLING_OF
\s(her|his|with) (\w*\s*(sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings)) (.*)	4	2	SIBLING_OF
(.*) (were|was) (his|her) (\w*\s*(sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings))	1	4	SIBLING_OF
\s(have|has|had) ((one|two|three|four|five|six|seven|eight|nine|ten|[0-9]+) \w*\s*(sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings)) \W (.*)	5	2	SIBLING_OF
\s(have|has|had) ((one|two|three|four|five|six|seven|eight|nine|ten|[0-9]+) \w*\s*(sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings)) (.*)	5	2	SIBLING_OF
\s(the (sister|brother|sibling)) of (.*)	3	1	SIBLING_OF
\s((one|two|three|four|five|six|seven|eight|nine|ten|[0-9]+) (sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings)) \W (.*)	4	1	SIBLING_OF
\s((one|two|three|four|five|six|seven|eight|nine|ten|[0-9]+) (sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings)) (.*)	4	1	SIBLING_OF
\s's (\w*\s*(sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings)) \W (.*)	3	1	SIBLING_OF
\s's (sister|sisters|brother|brothers|half-brother|half-brothers|half-sister|half-sisters|sibling|siblings) (.*)	2	1	SIBLING_OF

Table C.7: SIBLING_OF regular expressions

Regular Expression	Group No.	Relation
\s(co-founded|co-found|co-founder|co-founders|founder|co-creator|founding member|founding members|founding father) of (.*?) [\.,\/#!$%\^*:;{}=\-_~()]	2	FOUNDER_OF
\s(co-founded|co-found|co-founder|co-founders|founder|co-creator|founding member|founding members|founding father) .* of (.*?) [\.,\/#!$%\^*:;{}=\-_~()]	2	FOUNDER_OF
(.*) was (created|launched|established|formed|cofounded|co-founded|found|founded) (at|in|on|by)\s	1	FOUNDER_OF
\s(created|launched|established|formed|cofounded|co-founded|found|founding|founded|cofound|co-found|co-founder|co-founders|founder|founding|co-creator) ((?!by\s).*?) ([\.,\/#!$%\^*:;{}=\-_~()]|in |together with |with |and |from |to)	2	FOUNDER_OF

Table C.8: FOUNDER_OF regular expressions

Regular Expression	Group No.	Relation
\s(member|fellow|members|fellows) of (.*?) ([\.,\/#!$%\^*:;{}=\-_~()]|in |on |as |at |for |and)	2	MEMBER_OF
\s(admitted to|joined|join|joining) (.*?) ([\.,\/#!$%\^*:;{}=\-_~()]|in |on |as |at |for |and)	2	MEMBER_OF
\s(lead|leads|joins|joined|selected to|join|with) ((.*) (society|party|team|congress|movement|committee|association|squad|council|army|alliance))	2	MEMBER_OF
\s(for|of) ((.*?) (society|party|team|congress|movement|committee|association|squad|council|army|all (and){0,1}	2	MEMBER_OF

Table C.9: MEMBER_OF regular expressions

Regular Expression	Group No.	Relation
^the .* (child|son|daughter|children|sons|daughters|kids|boys|girls) of (.*)	2	CHILD_OF
^the (child|son|daughter) of (his|her) (father|mother) (.*)	4	CHILD_OF
^the \w* (child|son|daughter) of (his|her) (father|mother) (.*)	4	CHILD_OF
^the (child|son|daughter) of (.*)	2	CHILD_OF
^the \w* of (one|two|three|four|five|six|seven|eight|nine|ten|[0-9]+) (children|sons|daughters|kids|boys|girls|babies|twins) of (.*)	3	CHILD_OF
^the \w* (child|son}daughter) of (.*)	2	CHILD_OF
^(her|his) (father|mother|parent|parents) , (.*)	3	CHILD_OF
^(her|his) (father|mother|parent|parents) (.*)	3	CHILD_OF
^(her|his) \w* (father|mother|parent|parents) (.*)	2	CHILD_OF
^their (father|mother) , (.*)	2	CHILD_OF
^she is a daughter of (.*)	1	CHILD_OF
^he is a son of (.*)	1	CHILD_OF
\swas born .* to parents (.*)	1	CHILD_OF
\swas born .* to (.*)	1	CHILD_OF
\swas born to parents (.*)	1	CHILD_OF
\swas born to (.*)	1	CHILD_OF
\sborn to (.*)	1	CHILD_OF
\sthe (child|son|daughter) of (his|her) (father|mother) (.*)	4	CHILD_OF
\sthe \w* (child|son|daughter) of (his|her) (father|mother) (.*)	4	CHILD_OF
\sthe (child|son|daughter) of (.*)	2	CHILD_OF
\sthe \w* of (one|two|three|four|five|six|seven|eight|nine|ten|[0-9]+) (children|sons|daughters|kids|boys|girls|babies|twins) of (.*)	3	CHILD_OF
\sthe \w* (child|son}daughter) of (.*)	2	CHILD_OF
\sthe .* (son|daughter|children|sons|daughters|kids|boys|girls) of (.*)	2	CHILD_OF
\s(her|his) (father|mother|parent|parents) , (.*)	3	CHILD_OF
\s(her|his) (father|mother|parent|parents) (.*)	3	CHILD_OF
\s(her|his) \w* (father|mother|parent|parents) (.*)	3	CHILD_OF
(\w*) was (his|her) (father|mother|parent|parents)	1	CHILD_OF
\s(father|mother) , also called (.*)	2	CHILD_OF
\stheir (father|mother) , (.*)	2	CHILD_OF
\s's (father|mother) , (.*)	2	CHILD_OF
\s's (father|mother) was (.*)	2	CHILD_OF
\s's (father|mother) (.*)	2	CHILD_OF
\sis the .* children of (.*)	1	CHILD_OF
\sshe is a daughter of (.*)	1	CHILD_OF
\she is a son of (.*)	1	CHILD_OF

Table C.10: CHILD_OF regular expressions

Regular Expression	Group No 1	Group No. 2	Relation
\s((a|an) (first|second|third|fourth|fifth|sixth|seventh|eight|ninth|tenth) (child|son|daughter|kid|boy|girl|baby)) named (.*)	5	1	PARENT_OF
\s((first|second|third|fourth|fifth|sixth|seventh|eight|ninth|tenth) (child|son|daughter|kid|boy|girl|baby)) named (.*)	4	1	PARENT_OF
\s(a (child|son|daughter|kid|boy|girl|baby)) named (.*)	3	1	PARENT_OF
\s(one (child|son|daughter|kid|boy|girl|baby)) named (.*)	3	1	PARENT_OF
\s(child|son|daughter|kid|boy|girl|baby) named (.*)	2	1	PARENT_OF
\s((a|an) (first|second|third|fourth|fifth|sixth|seventh|eight|ninth|tenth) (child|son|daughter|kid|boy|girl|baby))\W* (.*)	5	1	PARENT_OF
\s(a (child|son|daughter|kid|boy|girl|baby))\W* (.*)	3	1	PARENT_OF
\s((one|two|three|four|five|six|seven|eight|nine|ten|[0-9]+) (child|son|daughter|kid|boy|girl|baby|twin|children|sons|daughters|kids|boys|girls|bab (.*)	4	1	PARENT_OF
\s((one|two|three|four|five|six|seven|eight|nine|ten|[0-9]+) \w* (child|son|daughter|kid|boy|girl|baby|twin|children|sons|daughters|kids|boys|girls|bab (.*)	4	1	PARENT_OF
\s(their|his|her) (child|son|daughter|kid|boy|girl|baby|twin|children|sons|daughters|ki (.*)	3	2	PARENT_OF
\s(their|his|her) ((first|second|third|fourth|fifth|sixth|seventh|eight|ninth|tenth) (child|son|daughter|kid|boy|girl|baby|twin|children|sons|daughters|kids|boys|girls|bab (.*)	5	2	PARENT_OF
\s(twin (children|sons|daughters|kids|boys|girls|babies))\W* (.*)	3	1	PARENT_OF

Table C.11: PARENT_OF regular expressions

Bibliography

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden markov support vector machines. In *ICML*, pages 3–10, 2003.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, March 1996. ISSN 0891-2017. URL `http://dl.acm.org/citation.cfm?id=234285.234289`.

John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, George Miller, Dan Moldovan, Bill Ogden, John Prager, Ellen Riloff, Amit Singhal, Rohini Shrihari, Tomek Strzalkowski, Ellen Voorhees, and Ralph Weischedel. Issues, tasks and program structures to roadmap research in question & answering (Q&A). Technical report, NIST, 2001. URL `http://www-nlpir.nist.gov/projects/duc/roadmapping.html`.

Grzegorz Chrupala and Dietrich Klakow. A named entity labeler for german: Exploiting wikipedia and distributional clusters. In Bente Maegaard Joseph Mariani Jan Odjik Stelios Piperidis Mike Rosner Daniel Tapias Nicoletta Calzolari (Conference Chair), Khalid Choukri, editor, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA). ISBN 2-9517408-6-7.

Joe Ellis. Tac kbp 2013 slot descriptions, 2013. URL `http://surdeanu.info/kbp2013/TAC_2013_KBP_Slot _Descriptions_1.0.pdf`.

David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79, 2010.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219885. URL `http://dx.doi.org/10.3115/1219840.1219885`.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Mach. Learn.*, 77(1):27–59, October 2009. ISSN 0885-6125. doi: 10.1007/s10994-009-5108-8. URL http://dx.doi.org/10.1007/s10994-009-5108-8.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655813.

Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1072228.1072378. URL http://dx.doi.org/10.3115/1072228.1072378.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972470.972475.

Bonan Min, Xiang Li, Ralph Grishman, and Sun Ang. New york university 2012 system for kbp slot filling. In *Proceedings of the Fifth Text Analysis Conference (TAC 2012)*. National Institute of Standards and Technology (NIST), November 2012. URL https://cs.nyu.edu/ min/.

Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. The structure and performance of an open-domain question answering system. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 563–570, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. doi: 10.3115/1075218.1075289. URL http://dx.doi.org/10.3115/1075218.1075289.

Naoaki Okazaki. Crfsuite: a fast implementation of conditional random fields (crfs), 2007. URL http://www.chokkan.org/software/crfsuite/.

L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. ISSN 0018-9219. doi: 10.1109/5.18626.

Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 147–155, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-29-9. URL http://dl.acm.org/citation.cfm?id=1596374.1596399.

Deepak Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 41–47, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073092. URL http://dx.doi.org/10.3115/1073083.1073092.

Benjamin Roth, Grzegorz Chrupala, Michael Wiegand, Singh Mittul, and Klakow Dietrich. Saarland university spoken language systems at the slot filling task of tac kbp 2012. In *Proceedings of the Fifth Text Analysis Conference (TAC 2012)*, Gaithersburg, Maryland, USA, November 2012. National Institute of Standards and Technology (NIST). to appear.

Robert F. Simmons, Sheldon Klein, and Keren McConlogue. Indexing and dependency logic for answering english questions. *American Documentation*, 15 (3):196–204, 1964. ISSN 1936-6108. doi: 10.1002/asi.5090150306. URL http://dx.doi.org/10.1002/asi.5090150306.

Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the conll-2000 shared task: chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning - Volume 7*, ConLL '00, pages 127–132, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. doi: 10.3115/1117601.1117631. URL http://dx.doi.org/10.3115/1117601.1117631.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073478. URL http://dx.doi.org/10.3115/1073445.1073478.