Knowledge graph construction for research literatures

Alisson Oldoni

A research project report submitted for
the degree of
Master of Computing and Information Technology

School of Computer Science and Engineering
The University of New South Wales

14 December 2011

Abstract

Contents

Contents **iv**

1 Introduction **1**

2 Information Extraction **5**
- 2.1 Natural Language Processing 5
- 2.2 Information Extraction 12
- 2.3 Knowledge Graphs 16

3 Challenges and Related Work **19**
- 3.1 Peculiarities with Academic Text 19
- 3.2 Related Work 20

4 Developed Workflow **21**
- 4.1 Tools 21
- 4.2 Developed process and programs 21

5 Results **23**

6 Conclusion and Future Work **25**

Bibliography **27**

Chapter 1

Introduction

Information extraction (IE) is the process of obtaining in an automatic fashion facts and information from unstructured text that can be read by a machine [15].

Historically, it mostly started with exercises on template filling based on raw natural text [21] as part of the Message Understanding Conferences (MUC) from the late 1980s and 1990s. As part of the MUC, competitions would take place in which a corpus would be made available of a specific domain, and different teams with different programs would try to extract the information from the natural text as to fill in the intended templates.

Note the following text from a news report regarding the result of a soccer match:

> '*Though Brazilian star Diego Tardelli's equaliser denied the Sky Blues victory at Jinan Olympic Sports Centre Stadium on Wednesday night, David Carney banked a precious away goal that will bode well for Graham Arnold's side when they host Shandong in next week's second round-of-16 leg. Sydney FC have taken a sizeable step towards a maiden Asian Champions League quarter-final berth after securing a 1-1 draw with Shandong Luneng in China.*'

```
Team 1:        ____________________
Team 2:        ____________________
Winner:        ____________________
Location:      ____________________
Final Score:   ____________________
```

Figure 1.1: An example of template to be filled in the sports domain.

An example of a task would be, to solely based on the above raw text, to fill in the template shown in Figure 1.1. The MUC competition would be based in various corpus and tasks based on varieties of news reports, such as satellite launches, plane crashes, joint ventures and other different data in these specific domains.

On the above example, one can observe that the *Team 1* is *Sydney FC*, *Team 2* is *Shandong Luneng*, there was no *Winner*, and consequently the *Location* and the *Final Score*. It gets more interesting as you observe the same type of information being delivered by a different reported:

> '*SYDNEY FC take the advantage of an away goal in China, leaving the second leg of their Asian Champions League Round of 16 tie with a 1-1 draw with Shandong Luneng.*'

Although roughly similar in this case, the approach to retrieve the data from Natural Language text needs to be able to generalise to the various ways a reporter might write such information. This effort becomes more complex as one moves through different domains and audiences of a text, such as: technical manuals, academic papers from different areas, legal text, contracts, financial news, biomedical, among others.

More recently, the output of such Information Extraction systems are used as to build other systems, more prominently Knowledge Graphs. A Knowledge Graph (KG), also known as the knowledge base, is a collection of the machine-readable database that contains entities, the attributes of entities and the relationships between entities [11]. Information Extraction tools would harvest data form unstructured or semi-structured text and provide such databases.

Popular search engines such as Google [11] and Bing [3] leverage Knowledge Graphs as to provide entity summary information and the related entities based on the query that the user is searching for. It is an essential foundation for many applications that requires machine understanding.

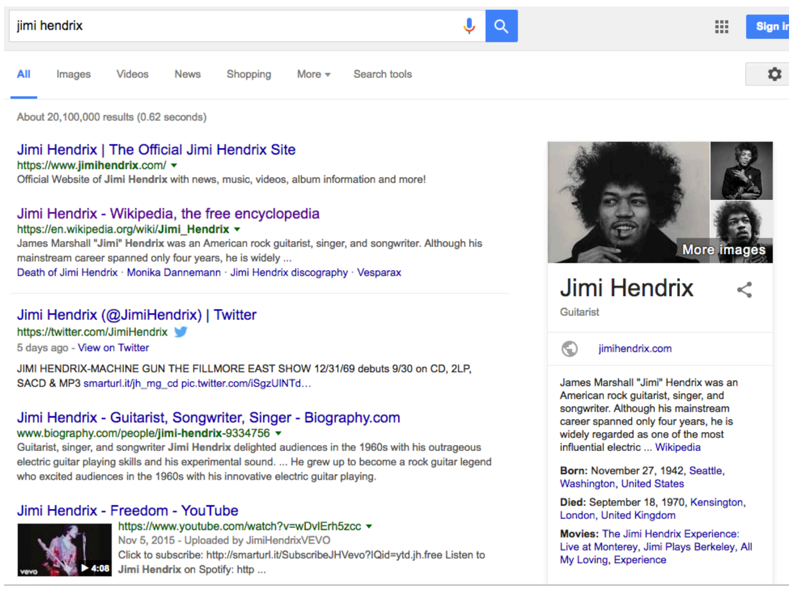
The use of Knowledge Graphs then allow users to be able to see extra information in a summarised table-like form, as to resolve their query without having to navigate to other sites. Note in the example in Figure 1.2 how the right column represents a sequence of facts of the *Jimi Hendrix* entity, in this case an entity of the class (or type) *person*, such as: his official website; where and when he was born; where and when he died; and a list of movies where this person is the subject of.

Modern pipelines for building Knowledge Graphs from raw text would then encompass several Information Extraction techniques, such as the ones below:

1. Discover entities in the text;

2. Discover relationships between these entities;

3. Perform entity disambiguation



About 20,100,000 results (0.62 seconds)

Jimi Hendrix | The Official Jimi Hendrix Site
https://www.jimihendrix.com/
Official Website of **Jimi Hendrix** with news, music, videos, album information and more!

Jimi Hendrix - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Jimi_Hendrix**
James Marshall "**Jimi**" **Hendrix** was an American rock guitarist, singer, and songwriter. Although his mainstream career spanned only four years, he is widely ...
Death of Jimi Hendrix · Monika Dannemann · Jimi Hendrix discography · Vesparax

Jimi Hendrix (@JimiHendrix) | Twitter
https://twitter.com/JimiHendrix
5 days ago - View on Twitter
JIMI HENDRIX-MACHINE GUN THE FILLMORE EAST SHOW 12/31/69 debuts 9/30 on CD, 2LP, SACD & MP3 smarturl.it/jh_mg_cd pic.twitter.com/iSgzUINTd...

Jimi Hendrix - Guitarist, Songwriter, Singer - Biography.com
www.biography.com/people/**jimi-hendrix**-9334756
Guitarist, singer, and songwriter **Jimi Hendrix** delighted audiences in the 1960s with his outrageous electric guitar playing skills and his experimental sound. ... He grew up to become a rock guitar legend who excited audiences in the 1960s with his innovative electric guitar playing.

Jimi Hendrix - Freedom - YouTube
https://www.youtube.com/watch?v=wDvlErh5zcc
Nov 5, 2015 - Uploaded by JimiHendrixVEVO
Click to subscribe: http://smarturl.it/SubscribeJHVevo?IQid=ytd.jh.free Listen to **Jimi Hendrix** on Spotify: http ...

Jimi Hendrix
Guitarist
jimihendrix.com
James Marshall "Jimi" Hendrix was an American rock guitarist, singer, and songwriter. Although his mainstream career spanned only four years, he is widely regarded as one of the most influential electric ... Wikipedia
Born: November 27, 1942, Seattle, Washington, United States
Died: September 18, 1970, Kensington, London, United Kingdom
Movies: The Jimi Hendrix Experience: Live at Monterey, Jimi Plays Berkeley, All My Loving, Experience

Figure 1.2: An example of knowledge graph application in the Google's result page.

4. Link entities to a reference Knowledge Graph (e.g., Yago2 [27] or DB-Pedia [16]).
5. Improve the quality of the output via input data cleaning, robust extraction, and learning-based post-processing methods;
6. Reason about how accurate these facts are;
7. Finally presenting the facts in a graph (the Knowledge Graph).

Some of these techniques will be explained further as part of this document.

In this project, we focus on studying and presenting some Information Extraction techniques as to build a domain-specific verb-centric information extraction tool that extracts relations from academic papers. More specifically, we focus on papers from the topic of databases and attempt to extract information from these papers for posterior usage by other systems with applications such as:

- Allow for structured and fast search of techniques in the papers and the possible relations between them;
- Possibly group papers by their used techniques;
- Discovery of techniques to improve performance on a certain problem;

- Generate a hierarchy of concepts, and their use;
- Among others.

An existing service that organises data from academic papers is the Semantic Scholar [25] project, by Professor Oren Etzioni from Allen Institute for AI. However, Semantic Scholar only understands a limited number of relationships (such as 'cite', 'comment', 'use_data_set', and 'has_caption') which are also more closely related to the meta-data about the paper, but not from the knowledge that the paper itself presents. Other similar services are Microsoft Academic Graph [19], Google Scholar [12], and CiteSeerX [7].

In the next sections, this document will give some background information on the techniques needed to achieve the above (Chapter 2), and it will also define the problem more precisely (Chapter 3), building as to introduce the development of this research (Chapter 4). In Chapter 5 we will describe some of the results, followed by the some final remarks in Chapter 6.

Chapter 2

Information Extraction

Information Extraction, a term already defined in the introduction, is a hard problem which mostly relies in attempting to use a computer to understand information explicitly stated in the form of natural language.

It is interesting to observe that, before writing down thoughts in a paper, an academic form the ideas of what facts he/she wants to express in his/hers head, and then attempt a structure to most clearly state these in text form. These multiple facts, and the relations between them, are then stated in a sentence in what is assumed to be a somewhat logical format, following the semantics of the language, whether English or any other. Following this example, one must then also assume that the future reader of this paper will use the reverse process to decode this information into facts or ideas to be understood. In fact, this assumption is what justifies the attempt of Information Extraction.

Several initiatives in the Natural Language Processing area attempt to understand and map what these semantic rules are, and how one could use a computer for tackling natural language related tasks. These initiatives are fruitful and provide advanced tools and techniques in which some will be described in this chapter.

2.1 Natural Language Processing

Natural Language Processing (or *NLP*) can be a term used to discuss any kind of computer manipulation of natural language text, also called raw text. It can mean simple things such as counting words and obtain their frequency distribution to compare different writing styles, or in a more complex sense it would require the understanding of human writings, to the extent of being able to extract information and meaning from it, or also give useful responses to them [4]. On this more complex end of the spectrum, where one wants to understand raw text, language technology and existing tools rely on formal models, or representations, of knowledge of language at the levels of morphology, syntax, semantics, among others linguistic concepts. A number of formal models including state machines, formal rule systems, logic, and probabilistic models are used to capture this knowledge from text and reason with it [15].

When information is laid out in natural language form, one start the analysis of the information presented by constructing a phrase or sentence based on smaller pieces of information such as verbs, nouns, and adjectives which are called the constituents. These then build up to form sequences of simple and complex sentences.

Observing more carefully a simple sentence, such as *'The police chased him.'*, it is possible to attempt to sample the different syntactic information presented in it As a first step it is possible to dissect its constituent parts as per Figure 2.1.

```
The/DT police/NN chased/VBD him/PRP ./.
```

Figure 2.1: An example of tagged sentence.

The first word in the sentence, *The*, is a *DT* or determiner. Other possible determiners include *my*, *your*, *his*, *her*. The second word is *police* is a *NN* which is the tag for a singular noun. With this information we can already tell that this sentence is speaking about something, and this something is the noun *police*. Subsequently the tag *VBD* is presented which specifically indicates the word *chased* is a verb (an action) in the past tense. At this point one can observe that something or someone (in this case the *police*) did something in the past.

This is already great information to have about the sentence. These tags that were added to the text in Figure 2.1 are called Part-Of-Speech tags, or POS tags [15]. The standardization of these tags and work to develop and tag existing text with them is done by the Penn Treebank project [18].

Note how specific the tags are, dictating the type of the word, a verb for an example, and its variation either in quantity or tense. Some other examples of these tags are shown in Table 2.1. Although most of them are simple to understand, note that *ADP* are the adpositions, which encompasses prepositions and postpositions. Some systems, such as the spaCy Natural Language Processing parser [14, 26] also maps these more specific tags into more general ones, for an example, while three different words in a sentence are tagged independently as *VBD*, *VBG* and *VBZ*, they are also tagged with a *VERB* tag. This is useful, if the user is not too interested in the detail of which verb variation was used.

A Part-Of-Speech Tagger is then a system that, given a raw text as input, assigns parts of speech to each word (or token) and is able to produce as output the tagged version of this text. The text in Figure 2.1 was tagged

POS tag	**Meaning**	**Sample**
ADP	Adpositions	*at*, or *in*
CONJ	Coordinating conjunction	*and*, or *or*
DT	Determiner	The
JJ	Adjective	She is *tired*
JJR	Adjective, comparative	That one is *larger*
JJS	Adjective, superlative	That is the *largest*
NN	Noun, singular or mass	Car
NNS	Noun, plural	Cars
NNP	Proper noun, singular	Microsoft
NNPS	Proper noun, plural	The *Kennedys*
RB	Adverb	She said *firmly*
VB	Verb, base form	Attack
VBD	Verb, past tense	Attacked
VBG	Verb, gerund or present participle	Attacking
VBN	Verb, past participle	Broken
VBP	Verb, non-3rd person singular present	I *attack*
VBZ	Verb, 3rd person singular present	He *attacks*

Table 2.1: List of some of the possible Part-Of-Speech (POS) tags.

using the Stanford Log-linear Part-Of-Speech Tagger [30].

POS Tag	**Word**	**Prev. Word**	**Prev. Tag**
DT	The	<START>	<START>
NN	police	The	DT
VBD	chased	police	NN
PRP	him	chased	VBD
.	.	him	PRP

Table 2.2: Features for sequential POS tagging.

The task of assigning these tags starts by deciding what are the tokens in a raw text sentence, and what are its sentences. As an example, the tokenizer needs to decide if a period symbol near a word represents an abbreviation (e.g. *Dr.*) or a sentence boundary - in case of an abbreviation, this period is then considered simply a token within the sentence. Another common problem in this step is deciding if a single quote is part of a word,(e.g. *It's*), or is delimiting a quoted part of the sentence, thus potentially hinting other semantic meanings. The Stanford POS Tagger used in this example also contain a tokenizer, which is part of the Stanford CoreNLP [17], a set of natural language analysis tools.

Modern POS Taggers tackle this task using a technique called Sequence Classification. A machine learning classifier model is then trained with a corpus of manually tagged text and has as an input certain features that might indicate which tag a token being currently analysed should be assigned with. Observing again the example from Figure 2.1, but now presented in table format in 2.2, it is easier to see how this learning algorithm would be trained to predict tags on unseen test. Note the second word *police*. For this word we are providing 4 features for the learning algorithm: the manually labelled POS Tag, the word itself, the previous word and the previous POS Tag. Suppose now that this sentence in included in a bigger corpus, and this pattern is a common one and the learning algorithm is provided with a substantial amount of labelled data in which this situation repeats itself: a *NN* is the second word in a setence, with *The* and *DT* being the previous word and tag respectively.

After the training, this model would behave in a similar fashion once presented with unseen data. Suppose now the first column on Table 2.2 is not presented. The model would pick the first word *The* and observe the features: *<START>* and *<START>* respectively and given that in our previously described corpus this is a common occurrence, it would then label this word with *DT*. Now, for the second word *police*, the features would be *The* for previous word, and *DT* for previous tag. Again, it is common for a noun to be placed after a determiner so the model assigns the label *NN* to the word *police*. The features used by the Sequence Classifies both while training and when using the model may vary and they impact the quality of the predictions it makes. The Stanford POS Tagger uses a broad use of lexical features, including jointly conditioning on multiple consecutive words [30].

A helpful concept at this stage is known as the lemma. A lemma is a canonical way of representing a word which strips out variations for quantity, tense, among others [15]. For an example, given the words *running* or *runs*, NLTK [4] outputs *run* as their lemma. The lemma can, for an example, be used together or instead of the existing features for training models.

```
Jetstar/NNP Airways/NNPS ,/PUNC a/DET unit/NN of/ADP Qantas/NNP
                  Airways/NNP Limited/NNP
```

Figure 2.2: An example of another tagged sentence.

When reading a text, it is also important to understand the relation between the words or sub-sentences that are contained in the phrase, and this is specially useful for information extraction. As an example, suppose the sentence *'Jetstar Airways, a unit of Qantas Airways Limited'*. There are different ways of breaking down the relationship of the words in this sentence.

Starting again with the POS tagging discussed earlier, one can see in 2.2 what are the types of the words that constitutes this sentence. As a further step, it's possible to see what are the sub-sentences or parts that form the phrase structure. The sub-sentences are connected upwards to one head (also called, *parent*), and downwards to one or more governors (also called *dependants*, or *children*), in a recursive structure. In Figure 2.3, the phrase *of Qantas Airways Limited*, which is part of the bigger phrase we are using as an example, is a prepositional phrase. A prepositional phrase lacks either a verb or a subject, and serve to assert binary relations between their heads and the constituents to which they are attached, in this case *a unit* [15]. The sub-sentence *a unit of Qantas Airways Limited* is then a noun phrase, since it contains and talks about a noun *unit*.

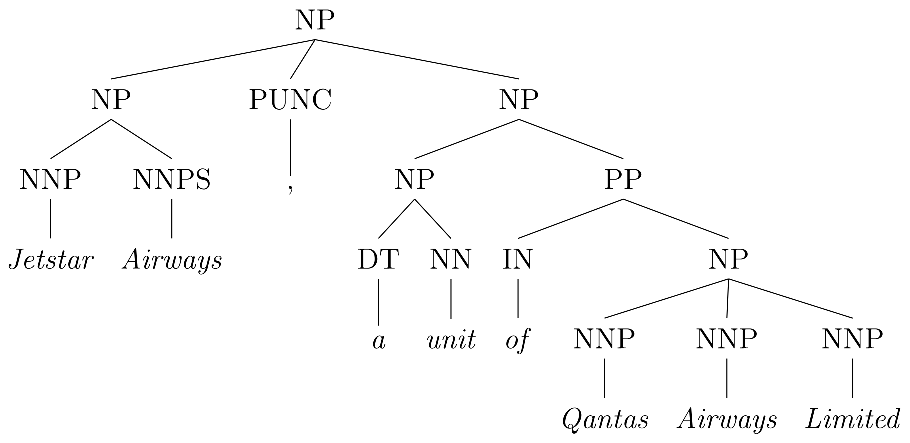


Figure 2.3: A sentence broken down to its Phrase Structure (sub-sentences).

After discovering the structure between the phrases and its sub-phrases, finding the syntactical dependency between words themselves is also a very interesting and useful task. Continuing on the same example, one can see how the sentence states the simple fact that one company (*Jetstar*) is a unit of another company (*Qantas*). The dependency tree in this case tells us that *Jetstar/NN* is the head of another noun *Airways* and that the relation between them is of the *compound* type. This relation is held between any noun that serves to modify the head noun and in this case indicate that this single entity is formed by two different words that are nouns. Note that *Jetstar* has no parents and thus is the root of the sentence. The dependencies can be fully viewed is Figure 2.4.

The next relation is the *appos* from *Airways* to *unit*. The appositional modifier relation indicates that the noun immediately to the right of the first noun that serves to define or modify the meaning of the former. One now already knows that *Jetstar Airways* is a *unit*, and, in the business context, it probably means that it is a company that belongs to a bigger company.

Continuing the analysis, the next relation is of the *prep* type and it indicates a prepositional modifier of a verb, adjective, or noun (our case), and

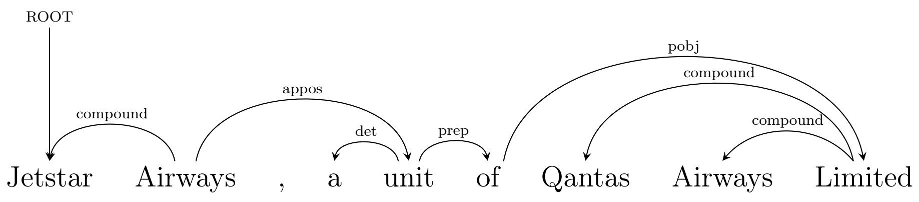


Figure 2.4: A sentence and the dependencies between the words.

it serves to modify the meaning of the verb, adjective, noun, or even another preposition. Note that this relation simply indicates the word pointed to by the edge is the preposition (in this case *of*). The next relation, *pobj*, indicates the actual object of the preposition and the noun phrase following the preposition and what it related to. This tree was created by the spaCy dependency parser [14, 26]. This same tree can be visualised in a more traditional tree structure in Figure 2.5. Although not in this example, another very common relation is the relative clause modifier *recmod* (or *relcl* in some notations) relation. A relative clause modifier of an NP (*Noun Phrase*) is a relative clause modifying the NP. The relation then in the tree points from the head noun of the NP to the head of the relative clause, normally a verb. The explanations for the cited relations in this document were obtained from the Stanford typed dependencies manual [9], and the Universal Dependencies (UD) project [24], both which are reference for the possible relation and contain a full lists of their meanings.

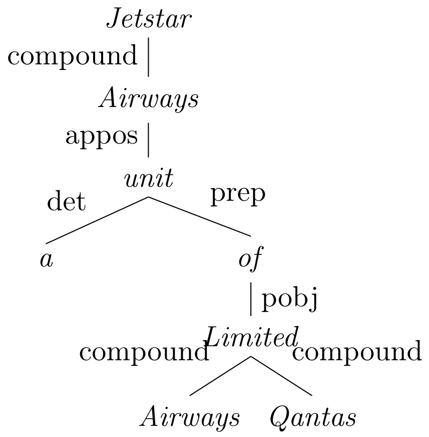


Figure 2.5: A sentence and the dependency tree, showing the syntactical relation between these words.

The software that is able to output a syntactical dependency tree, given a sentence, is called a dependency parser. Several different methods can be used to achieve this. One way is by defining dependency grammars, and then parsing text using these grammar. The grammar would contain words and its possible heads, and it would be applied repeatedly into the text in a process called cascaded chunking [4]. More recent methods use a process called Shift-reduce, in which the sentence is kept in a queue with the left-most token in front of the queue. The model could then decide between applying 3 operations:

1. Shift: move one token from the queue to stack.

2. Reduce left: top word on stack is head of second word.

3. Reduce right: second word on stack is head of top word.

A model is then trained to predict, given a text that is added to the queue, what is the next move that it should take, and what is the sequence of moves that will result in the best possible final dependency tree. This is done in a monotonic manner, in the sense that once a decision is made by the model, it cannot change it. Full working examples of this method are described in [6]. Other methods also use these 3 possible decisions, but allow the parser to be non-monotonic and go back in the tree and change previous decisions given new evidence form the features, such as spaCy and its dependency parser [14, 26]. Another method also uses the same set of decisions, but does a beam-search observing multiple partial hypotheses and keeping them at each step, with hypotheses only being discarded when there are several other higher-ranked hypotheses under consideration, such as the Syntaxnet parser [29, 2]. All these cited models use neural networks as the method to form the model.

As a further example, besides providing dependency parsing tree, spaCy also provides an iterator so you can obtain what is called the noun chunks of the document. The noun chunks are smaller pieces of the sentences within the document that are base noun phrases, or a 'NP chunk' (as per Figure 2.3). They are noun phrases that do not permit other NPs to be nested within it so no NP-level coordination (e.g.: *'cat/NN and/CONJ dog/NN'*), no prepositional phrases, and no relative clauses [26].

Other problems tackled by the Natural Language Processing discipline and that are relevant for Information Extraction are *Coreference resolution* and *pronominal anaphora resolution*. Coreference resolution intends to define all possible entities that a text can reference to in some sort of definitive list, or more precisely a *discourse model*, find in the text all the chained references to these entities, and link them to the specific entities. While very similar in nature, pronominal anaphora resolution is more simple as it is the problem of resolving in a given sentence to which previous NN (noun) or NNP (proper noun) a single PRP (pronoun) refers to [15].

Take for an example the sentence *'John is a quiet guy, but today he is furious.'*, the initial mention of the entity *John* appears in the first token.

Token 5 talks about a *guy*, which although not a pronoun, is still a reference to the previous entity in the first word. The ninth token is a pronoun and again refers to the same *John*, so is part of the chain of mentions. The full resolution chain would be denoted in Figure 2.6.

Normally these systems work towards analysing pairs of tokens using a probabilistic model, and then decide how likely they are references for the same entity. More recent approaches also group possible tokens in a cluster and use cluster-level features to determine the chains of coreferences [8]. The tool used to extract is the Stanford Coreference Resolution annotator [8], which is part of the latter group of tools that use cluster level features. This annotator is also part of the CoreNLP toolset [17].

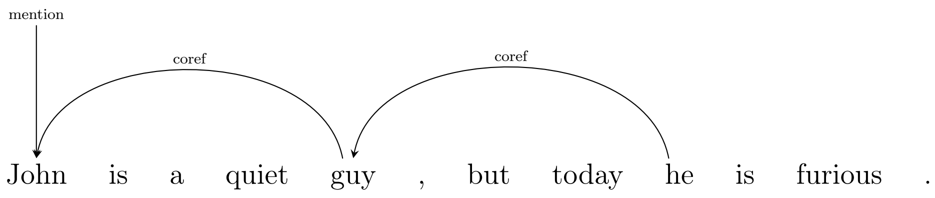


Figure 2.6: A sentence, the mentions of an entity, and the proposed coreference resolutions.

All linguistic data mentioned in this section is data that can be obtained from the raw text itself, and is then the base for several features in which the classifiers for Information Extraction act on.

2.2 Information Extraction

Also according to [15], the IE process is, in general, divided in the following subtasks: Named Entity Recognition (NER), Coreference Resolution, Entity Disambiguation, Relation Extraction (RE), Event Detection, Temporal Analysis. The main subtasks relevant to this report will be described further in this section.

Once the information is extracted it is then used for tasks such as Template Filling [15], or stored as a Knowledge Graph for downstream logical reasoning or for further queries.

[$_{\text{PER}}$ James Cook] was born on 27 October 1728 in the village of [$_{\text{LOC}}$ Marton] in [$_{\text{COUNTY}}$ Yorkshire].

Table 2.3: An example of Named Entity Recognition (NER).

Name Entity Recognition is the process of, given a sentence, mark what are the entities that are part of it. Once the entity is detected, it needs to be classified within the classes of the given domain - in the spirit of the previous examples this would be e.g.: *City* or *Person.* A few approaches exist for the problem of NER, mostly related to Pattern Matching or Sequence Classification.

Pattern	**Would yield ENTITY of type**
[PERSON] was born	PERSON
in the village of [LOC]	LOCATION
in [LOC]	LOCATION

Table 2.4: Examples of Named Entity Recognition (NER) patterns, based on the sentence from Table 2.3.

Observe, for an example, the sentence in Table 2.3. Several articles regarding prominent figures, either historical or of our current society, can start with the text *[ENTITY] was born.* One approach might be Pattern Matching, which is to mine the input natural language text while looking for this pattern using Regular Expressions - or a Finite-State Automata [15]. The entities found by this pattern would then also receive the *person* class. Other possible patterns can be found in Table 2.4.

Another way to extract entities from text is to frame the NER problem as a Sequence Classification problem. It requires the training of a classifier in which, given the class of the previous word, and other surrounding features of the current word, will attempt to guess if the current word is an entity, and if it is, also guesses its class.

To achieve this, previously annotated data with existing sentences and its entities is needed. The format in which this annotated data is provided varies, however more commonly the IOB format (Table 2.5) is used in several of the NER tools, including NLTK [4] and the popular Stanford Named Entity Recognizer (NER), part of the Stanford CoreNLP [17]. Stanford CoreNLP provides a set of natural language analysis tools.

The IOB format also helps remove ambiguity in case there are two contiguous entities of same class without any word tagged as *O* in between. In practice, these cases are somewhat rare in several domains, and thus a simplified version without the *B-* and *I-* prefixes are used [28].

The Stanford Named Entity Recognizer (NER), also known as CRF-Classifier [10], provides a general implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models. A CRF is a conditional sequence model which represents the probability of a hidden state sequence given some observations.

Several useful features can be used during the training of a NER CRF classifier model. In Table 2.6, several examples are presented. The Word Shape feature is an interesting addition from recent research, as it captures the notion that most entities are written in capital letters, or starting with

Word	**Tag**
James	B-PERSON
Cook	I-PERSON
was	O
born	O
on	O
27	B-DATE
October	I-DATE
1728	I-DATE
in	O
the	O
village	O
of	O
Marton	B-LOC
in	O
Yorkshire	B-LOC
.	O

Table 2.5: Example of IOB-formatted sentence used to train classifiers for the Named Entity Recognition (NER) task, based on the sentence from Table 2.3.

capital letter, or containing numbers in the middle of the word, and other specific shapes.

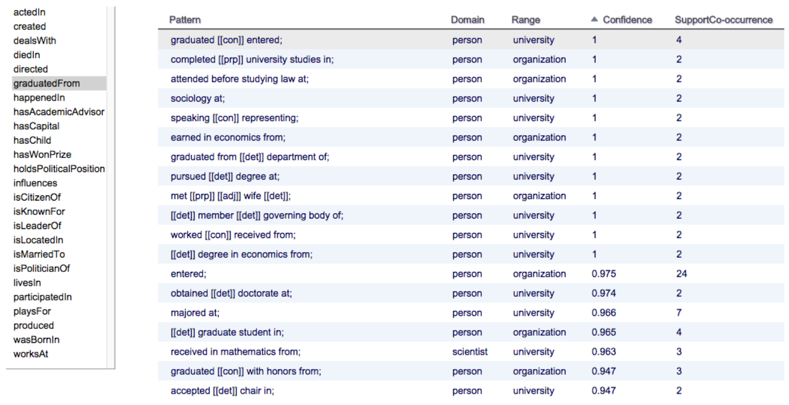
Feature	**Description**
Word	The current word being classified.
N-grams	A feature from n-grams, i.e., sub-strings of the word.
Previous Class	The class of the immediate previous word.
Previous Word	The previous word.
Disjunctive	Disjunctions of words anywhere in the left or right.
Word Shape	The shape of the word being processed captured using. In general replaces numbers with d, x to lower-case letters, and X to upper-case letters.

Table 2.6: Examples of features used to train the CRFClassifier.

In addition to the above methods another useful technique is the use of gazetteers. Gazetteers are common for geographical data, where government provided lists of names can contain millions of entries names for all manner of locations along with detailed geographical, geologic and political information [15].

Relation Extraction (RE) is the ability to discern the relationships that exist among the entities detected in a text [15], and is naturally the next challenge after being able to detect entities. It can be done using Pattern Matching, Classifiers, or purely by exploiting linguistic data available form a sentence.

The Pattern Matching technique from NER is then built upon in the Relation Extraction step, and now involves more than one entity, yielding binary relations. This approach is used in tools such as PROSPERA [22] or those mined by PATTY [23]. More specifically, examples of patterns mined by PATTY for the *graduatedFrom* relation are seen in Figure 2.7.



Pattern	Domain	Range	▲ Confidence	SupportCo-occurrence
graduated [[con]] entered;	person	university	1	4
completed [[prp]] university studies in;	person	organization	1	2
attended before studying law at;	person	organization	1	2
sociology at;	person	university	1	2
speaking [[con]] representing;	person	university	1	2
earned in economics from;	person	organization	1	2
graduated from [[det]] department of;	person	university	1	2
pursued [[det]] degree at;	person	university	1	2
met [[prp]] [[adj]] wife [[det]];	person	organization	1	2
[[det]] member [[det]] governing body of;	person	university	1	2
worked [[con]] received from;	person	university	1	2
[[det]] degree in economics from;	person	university	1	2
entered;	person	organization	0.975	24
obtained [[det]] doctorate at;	person	university	0.974	2
majored at;	person	university	0.966	7
[[det]] graduate student in;	person	organization	0.965	4
received in mathematics from;	scientist	university	0.963	3
graduated [[con]] with honors from;	person	organization	0.947	3
accepted [[det]] chair in;	person	university	0.947	2

Figure 2.7: An example of patterns extracted from PATTY for the *graduatedFrom* relation.

PROSPERA's main technique is that not only it obtain facts based on a small set of initial seed patterns, but also obtain new candidate patterns based on the mined facts. Once the process finishes, these new candidate patterns are evaluated and then added to the the existing pattern repository for re-use. The whole process then iterates again finding even more facts from these new patterns, and new candidate patterns.

Another tool in the Stanford CoreNLP package, the Relation Extractor [28] is a classifier to predict relations in sentences.

The RMD model was built from scratch as a multi-class classifier that extracts binary relations between entity mentions in the same sentence. During training, known relation mentions become pos- itive examples for the corresponding label and all other possible combinations between entity mentions in the same sentence become negative exam- ples. We used a multiclass logistic regression classi- fier with L2 regularization. Our feature set is taken from (Yao et al., 2010; Mintz et al., 2009; Roth and Yih, 2007; Surdeanu and Ciaramita, 2007) and mod- els the relation arguments, the surface distance be- tween the relation arguments, and the syntactic path between the two arguments, using both constituency and dependency representations. For syntactic in- formation, we used the Stanford parser (Klein and Manning, 2003) and the Stanford dependency repre- sentation (de Marneffe et al., 2006). For RMD, we implemented an additive feature se- lection algorithm similar to the one in (Surdeanu et al., 2008), which iteratively adds the feature with the highest improvement in F1 score to the current feature set, until no improvement is seen. The algorithm was configured to select features that yielded the best combined performance on the dataset from Roth and Yih (2007) and the training partition of ACE 2007.4

- Classification
- Possible features
- Open Information Extraction
- Evaluation
- What drives the field forward

To be completed: entity disambiguation?

2.3 Knowledge Graphs

Knowledge Graphs contain a valuable of information in a structured format, traditionally originally mined from table-like structures form places like Wikipedia [31] tables [16]. It can be used for a diverse range of applications, such as helping other systems reason about quality of harvested facts[27], provide table-like facts about an entity[11], and question-answering systems[13]. Moreover, recent years have witnessed a surge in large scale knowledge graphs, such as DBpedia [16], Freebase [5], Googles Knowledge Graph [11], and YAGO [27].

The Knowledge Graph name follows from the data structure that is created from the facts in its final form, a graph with nodes representing entities and edges representing various relations between entities. In Figure 2.8, it is possible to observe an example plotted in this form. The list of possible entities classes, and allowable relations between entities is known as a schema. The schema represented in Figure 2.8 is detailed in Table 2.8; one can observe that, as an example, *Max Planck* is an entity of the type *physicist*.

```
type(A, D) :- type(A, B), subclassOf(B, C), subclassOf(C, D)
```

Table 2.7: This entailment example allows one to assert that `type(Max Planck, person)` is also true, based on the fact tuples presented in Table 2.8.

Moreover, based on the facts presented, entailments can be made and one trivial example is denoted in Table 2.7. More complex examples of possible reasoning can be seen in [28]. This is equivalent to traversing the graph from a node that represents a more specific information, to a node that represents a more general information - e.g.: another possible child node of *scientist* could be the type *biologist*.

```
type(Max Planck, physicist)
subclassOf(physicist, scientist)
subclassOf(scientist, person)
bornIn(Max Planck, Kiel, 1858)
type(Kiel, city)
locatedIn(Kiel, Germany)
hasWon(Max Planck, Nobel Prize, 1919)
```

Table 2.8: Some facts regarding Max Planck, also depicted in Figure 2.8.

This example denotes a classical domain, more precisely important persons, companies, locations, and the relations between them, in which Information Extraction (IE) tools have been very successful on.

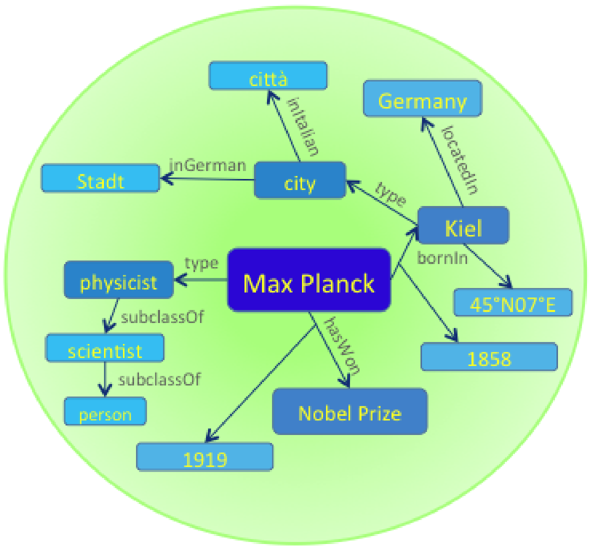


Figure 2.8: An example of knowledge graph from [27] plotted with vertices and edges.

As mentioned previously, YAGO [27] is a prominent Knowledge Graph database, and possesses several advanced characteristics. Every relation in its database is annotated with its confidence value. See the example of the resulting graph in Figure 2.9. Moreover, YAGO combines the provided taxonomy with WordNet [20] and with the Wikipedia category system [31], assigning the entities to more than 350,000 classes. This allow for very powerful querying. Finally, it attaches a temporal and a spacial dimension to many of its facts and entities, being then capable to answer questions such as *when* and *where* such event took place.

WordNet is a semantically-oriented dictionary of English, similar to a traditional thesaurus but with a richer structure [4]. More specifically, it provides relations to synonyms, hypernyms and hyponyms, among others.

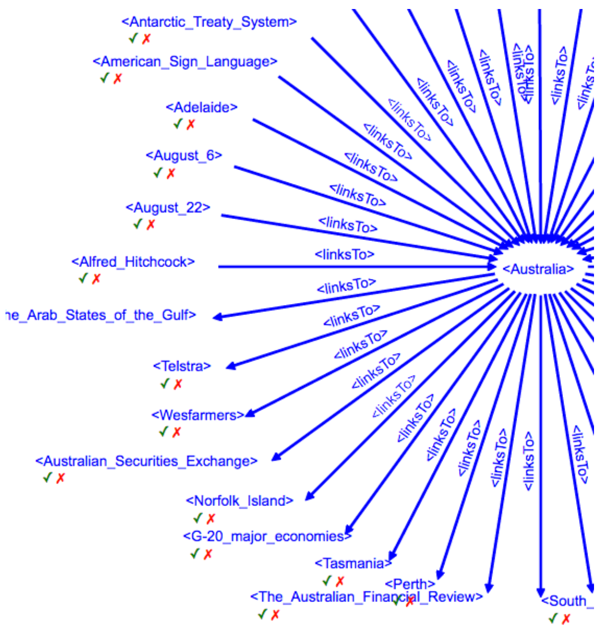


Figure 2.9: An example of patterns existants in YAGO.

Chapter 3

Challenges and Related Work

Problem accross domains, strict language vs. creative language.

Sarcasm.

We are extracting explicit information, and not trying to infer implicit information. That would be a different logical problem that would need to reason about using a database of information to compare what is being learned from the natural text with.

3.1 Peculiarities with Academic Text

Difficult to manually tag.

Existing tools that are based in models don't come equiped to predict relation in this kind of text. E.g.:

Relations that could be extracted, e.g. denes, don't appear with relevant Entities.

A lot of coreference problems, e.g. "their work". Even when their work is simply a paper reference - with unclear. One could trivially parse the above reference with the actual Entity name of the system or algorithm or technique elabrated in the reference paper as to mine the relations between the proper entities.

Methods:

* Use semi-automatic methods to collect research literature and convert them into plain text with certain markups. * Use existing open source solutions to parse the store the input data. * Build a pipeline to extract entities and relationship from the input data. * Perform entity disambiguation and linking to a reference Knowledge Graph (e.g., Yago2 or DBPedia). * Design effective postprocessing methods to improve the quality of the extraction. * Evaluate the entire extraction system.

3.2 Related Work

Other methods for relation extraction such as the ones from papers:
ClausIE: Clause-Based Open Information Extraction
Extreme Extraction: Only One Hour per Relation
IKE - An Interactive Tool for Knowledge Extraction

Chapter 4

Developed Workflow

The first step to in achieving our goal is obtaining the raw natural language text from papers in the area. Most research products such as thesis, papers, or any other report, are mostly available primarily in the Portable Document Format (PDF) format [1] - which then needs to then be parsed into a raw text in an automated manner.

4.1 Tools

Python

Java

requests

BeautifulSoup4

PDF extraction generates noisy output

NLTK

Brat

standoff2conll

corpkit

To be completed: Querying a corpus? How to check for similar words statistically, etc

corenlp-xml

Parsey

Spacey

Stanford CoreNLP

Tregex

Graphviz

4.2 Developed process and programs

We leverage on existing pipeline for NER and Relatino Extraction (Stanford) , instead of independent tools.

We use IO notatoion instead of IOB notation notation3 for entity mention la- bels (e.g., the labels for the tokens over the Seattle Seahawks on Sunday (from Figure 1) are encoded as O O NFLTEAM NFLTEAM O DATE). The IO notation facilitates faster inference than the IOB or IOB2 notations with minimal impact on performance, when there are fewer adjacent mentions with the same type. AS PER STANFORD RELATION EXTRACTOR PAPER.

TALK ABOUT FEATURES WE PICKED FOR NER.

TALK ABOUT FEATURES WE PICKED FOR REL.

Open information extraction (open IE) has been shown to be useful in a number of NLP tasks, such as question answering (Fader et al., 2014), relation extraction (Soderland et al., 2010), and infor- mation retrieval (Etzioni, 2011).

We did not implement coreference.

Expected Outcome:

1. A system that can build a knowledge graph from research literatures. 2. A written report about the detailed designs and implementations of this system. 3. A seminar to present the process and outcome of this project.

Chapter 5

Results

Chapter 6

Conclusion and Future Work

Furhter ideas: - Research relations through time. You could have a certain feature. - Reinforcements made to definitions in other papers. - Events, such as changes in conclusions - e.g.: this was the better technique, now this other technique is the best.

Bibliography

[1] *Adobe: What is PDF?* https://acrobat.adobe.com/us/en/why-adobe/about-adobe-pdf.html. [Online; accessed 14-August-2016]. 2016.

[2] Daniel Andor et al. 'Globally Normalized Transition-Based Neural Networks'. In: *CoRR* abs/1603.06042 (2016).

[3] *Bing Knowledge and Action Graph*. https://www.bing.com/partners/knowledgegraph. [Online; accessed 14-August-2016]. 2016.

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

[5] Kurt Bollacker et al. 'Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge'. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 1247–1250.

[6] Danqi Chen and Christopher Manning. 'A Fast and Accurate Dependency Parser using Neural Networks'. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 740–750.

[7] *CiteSeerX*. http://csxstatic.ist.psu.edu/about. [Online; accessed 27-September-2016]. 2016.

[8] Kevin Clark and Christopher D. Manning. 'Entity-Centric Coreference Resolution with Model Stacking'. In: *Association for Computational Linguistics (ACL)*. 2015.

[9] Marie catherine De Marneffe and Christopher D. Manning. *Stanford typed dependencies manual*. 2008.

[10] 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. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, pp. 363–370.

[11] *Google Knowledge Graph*. https://www.google.com/insidesearch/features/search/knowledge.html. [Online; accessed 14-August-2016]. 2016.

[12] *Google Scholar*. https://scholar.google.com/intl/en/scholar/about.html. [Online; accessed 27-September-2016]. 2016.

[13] Ben Hixon, Peter Clark and Hannaneh Hajishirzi. 'Learning Knowledge Graphs for Question Answering through Conversational Dialog'. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, 2015, pp. 851–861.

[14] Matthew Honnibal and Mark Johnson. 'An Improved Non-monotonic Transition System for Dependency Parsing'. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 1373–1378. URL: https://aclweb.org/anthology/D/D15/D15-1162.

[15] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

[16] Jens Lehmann et al. 'DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia'. In: *Semantic Web Journal* 6.2 (2015), pp. 167–195.

[17] Christopher D. Manning et al. 'The Stanford CoreNLP Natural Language Processing Toolkit'. In: *Association for Computational Linguistics (ACL) System Demonstrations*. 2014, pp. 55–60.

[18] Mitchell P. Marcus, Mary Ann Marcinkiewicz and Beatrice Santorini. 'Building a Large Annotated Corpus of English: The Penn Treebank'. In: *Comput. Linguist.* 19.2 (June 1993), pp. 313–330.

[19] *Microsoft Academic Graph*. https://www.microsoft.com/cognitive-services/en-us/academic-knowledge-api. [Online; accessed 27-September-2016]. 2016.

[20] George A. Miller. 'WordNet: A Lexical Database for English'. In: *Commun. ACM* 38.11 (Nov. 1995), pp. 39–41. ISSN: 0001-0782.

[21] Marie-Francine Moens. *Information Extraction: Algorithms and Prospects in a Retrieval Context (The Information Retrieval Series)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[22] Ndapandula Nakashole, Martin Theobald and Gerhard Weikum. 'Scalable Knowledge Harvesting with High Precision and High Recall'. In: *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*. WSDM '11. New York, NY, USA: ACM, 2011, pp. 227–236.

[23] Ndapandula Nakashole, Gerhard Weikum and Fabian Suchanek. 'PATTY: A Taxonomy of Relational Patterns with Semantic Types'. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. EMNLP-CoNLL '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 1135–1145.

[24] Joakim Nivre et al. *Universal Dependencies 1.3*. http://universaldependencies.org/. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague. 2016.

[25] *Semantic Scholar*. http://allenai.org/semantic-scholar/. [Online; accessed 20-August-2016]. 2016.

[26] *spaCy: Industrial-strength Natural Language Processing*. https://spacy.io/. [Online; accessed 27-September-2016]. 2016.

[27] Fabian M. Suchanek, Gjergji Kasneci and Gerhard Weikum. 'Yago: A Core of Semantic Knowledge'. In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. New York, NY, USA: ACM, 2007, pp. 697–706.

[28] Mihai Surdeanu et al. 'Customizing an Information Extraction System to a New Domain'. In: *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*. RELMS '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 2–10.

[29] *SyntaxNet: Neural Models of Syntax*. https://github.com/tensorflow/models/tree/master/syntaxnet. [Online; accessed 01-October-2016]. 2016.

[30] Kristina Toutanova et al. '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. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 173–180.

[31] *Wikipedia, The Free Encyclopedia*. http://www.wikipedia.org/. [Online; accessed 21-August-2016]. 2010.