A Swedish Natural Language Processing Pipeline For Building Knowledge Graphs

ALEJANDRO GONZÁLEZ PÉREZ
A Swedish Natural Language Processing Pipeline For Building Knowledge Graphs

MASTER THESIS

EIT ICT Innovation and Data Science

Author: Alejandro González Pérez
Supervisor: Thomas Sjöland  Examiner: Anne Håkansson
Course 2017-2018
To my parents. For their love, endless support and encouragement.
I would like to express my very great appreciation to my colleagues Anders Baer and Maria Jernstöm for their valuable and constructive suggestions during the planning and development of this research work. Their willingness to give their time so generously has been very much appreciated.

I would like also to express my deep gratitude to Anne Håkansson and Thomas Sjöland, my research supervisors, for their patient guidance, enthusiastic encouragement and useful critiques of this research work.
Abstract

The concept of knowledge is proper only to the human being thanks to the faculty of understanding. The immaterial concepts, independent of the material causes of the experience constitute an evident proof of the existence of the rational soul that makes the human being a spiritual being” in a way independent of the material.

Nowadays research efforts in the field of Artificial Intelligence are trying to mimic this human capacity using computers by means of teaching them how to read and understand human language using Machine Learning techniques related to the processing of human language. However, there are still a significant number of challenges such as how to represent this knowledge so can be used by a machine to infer conclusions or provide answers.

This thesis presents a Natural Language Processing pipeline that is capable of building a knowledge representation of the information contained in Swedish human-generated text. The result is a system that, given Swedish text in its raw format, builds a representation in the form of a Knowledge Graph of the knowledge or information contained in that text.
Abstract

Vetskapen om kunskap är den del av det som definierar den nutida människan (som vet, att hon vet). De immateriella begreppen oberoende av materiella attribut är en del av beviset på att människan en själsig varelse som till viss del är oberoende av materialet.

För närvarande försöker forskningsinsatser inom artificiell intelligens efterlikna det mänskliga betandet med hjälp av datorer genom att ”lära” dem hur man läser och förstår mänskligt språk genom att använda maskininlärningstekniker relaterade till behandling av mänskligt språk. Det finns emellertid fortfarande ett betydande antal utmaningar, till exempel hur man representerar denna kunskap så att den kan användas av en maskin för att dra slutsatser eller ge svar utifrån detta.

Denna avhandling presenterar en studie i användningen av ”Natural Language Processing” i en pipeline som kan generera en kunskapsrepresentation av informationen utifrån det svenska språket som bas. Resultatet är ett system som, med svensk text i råformat, bygger en representation i form av en kunskapsgraf av kunskapen eller informationen i den texten.
## Contents

<table>
<thead>
<tr>
<th>Contents</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
</tbody>
</table>

### 1 Introduction

- 1.1 Background .......................... 2
- 1.2 Problem ............................ 3
- 1.3 Purpose ............................ 4
- 1.4 Goal ................................ 4
  - 1.4.1 Benefits, Ethics and Sustainability .... 4
- 1.5 Research Methodology ............... 5
  - 1.5.1 Philosophical Assumption ............ 5
  - 1.5.2 Research Method ................... 5
  - 1.5.3 Research Approach ................. 5
- 1.6 Delimitations ...................... 6
- 1.7 Outline ............................ 6

### 2 Framework

- 2.1 Natural Language ................... 7
- 2.2 Swedish Language ................... 8
  - 2.2.1 Alphabet ........................ 9
  - 2.2.2 Nouns ........................... 9
  - 2.2.3 Adjectives ....................... 9
  - 2.2.4 Adverbs .......................... 10
  - 2.2.5 Numbers .......................... 10
  - 2.2.6 Pronouns ......................... 10
  - 2.2.7 Prepositions ...................... 11
  - 2.2.8 Verbs ............................ 11
  - 2.2.9 Conjunctions ...................... 11
  - 2.2.10 Syntax .......................... 12
- 2.3 Machine Learning ................... 12
- 2.4 Natural Language Processing ....... 13
  - 2.4.1 Word and sentence tokenisation .... 13
  - 2.4.2 Part-Of-Speech tagging ............. 14
  - 2.4.3 Syntactic Dependency Parsing ...... 14
  - 2.4.4 Named Entity Recognition .......... 15
  - 2.4.5 Relation and Information Extraction 16
- 2.5 NLP tools and frameworks .......... 16
  - 2.5.1 Stanford CoreNLP ................. 17
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5.2 NLTK Python Library</td>
<td>17</td>
</tr>
<tr>
<td>2.5.3 spaCy</td>
<td>17</td>
</tr>
<tr>
<td>2.5.4 SyntaxNet</td>
<td>18</td>
</tr>
<tr>
<td>2.6 Knowledge Representation</td>
<td>18</td>
</tr>
<tr>
<td>2.7 Graph databases and Knowledge graphs</td>
<td>19</td>
</tr>
<tr>
<td>3 Related Work</td>
<td>21</td>
</tr>
<tr>
<td>3.1 Information and Knowledge extraction</td>
<td>21</td>
</tr>
<tr>
<td>3.2 Knowledge representation</td>
<td>22</td>
</tr>
<tr>
<td>3.3 Machine Learning &amp; Natural Language Processing</td>
<td>22</td>
</tr>
<tr>
<td>4 Methods</td>
<td>24</td>
</tr>
<tr>
<td>4.1 Data collection</td>
<td>24</td>
</tr>
<tr>
<td>4.2 Data analysis</td>
<td>25</td>
</tr>
<tr>
<td>4.3 Data verification and validation</td>
<td>26</td>
</tr>
<tr>
<td>4.4 Software development</td>
<td>27</td>
</tr>
<tr>
<td>4.4.1 Software development stages</td>
<td>28</td>
</tr>
<tr>
<td>4.4.2 Software development models</td>
<td>28</td>
</tr>
<tr>
<td>5 The NLP Pipeline</td>
<td>31</td>
</tr>
<tr>
<td>5.1 Design</td>
<td>31</td>
</tr>
<tr>
<td>5.2 Generating information triples</td>
<td>33</td>
</tr>
<tr>
<td>5.2.1 Text preprocessing</td>
<td>33</td>
</tr>
<tr>
<td>5.2.2 Part-Of-Speech tagging</td>
<td>36</td>
</tr>
<tr>
<td>5.2.3 Syntactic dependency parsing</td>
<td>37</td>
</tr>
<tr>
<td>5.2.4 Named entity recognition</td>
<td>41</td>
</tr>
<tr>
<td>5.2.5 Relation extraction</td>
<td>43</td>
</tr>
<tr>
<td>5.3 Integrating information triples</td>
<td>43</td>
</tr>
<tr>
<td>5.4 Constructing the knowledge graph</td>
<td>44</td>
</tr>
<tr>
<td>6 Evaluation</td>
<td>47</td>
</tr>
<tr>
<td>7 Conclusions and Future Work</td>
<td>49</td>
</tr>
<tr>
<td>7.1 Conclusion</td>
<td>49</td>
</tr>
<tr>
<td>7.2 Discussion</td>
<td>50</td>
</tr>
<tr>
<td>7.3 Future work</td>
<td>52</td>
</tr>
<tr>
<td>Bibliography</td>
<td>53</td>
</tr>
<tr>
<td>Appendix</td>
<td></td>
</tr>
<tr>
<td>A Graphs</td>
<td>65</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Sample Swedish sentence structure analysis [31] . . . . 12
2.2 Grammatical Dependency Graph. . . . . . . . . . . . . . . . . . 15
2.3 Information Extraction example. Source: nlp.stanford.edu 16
2.4 NLP tools and frameworks accuracy evaluation [73]. . . 17
2.5 Graph database example. . . . . . . . . . . . . . . . . . . . . . . . . 19
2.6 Knowledge graph example. . . . . . . . . . . . . . . . . . . . . . . . 20
4.1 Grammar analysis. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
5.1 Main tasks identified. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
5.2 Sample triple extracted. . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
5.3 Sample triple extracted. . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
5.4 Constructed knowledge graph. . . . . . . . . . . . . . . . . . . . . . . 33
5.5 Format of the triples. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
5.6 Classification of data quality problems [132]. . . . . . . . . . . 34
5.7 Text transformations before linguistic analysis [135]. . . . . . 35
5.8 Text adjacency graph example. . . . . . . . . . . . . . . . . . . . . . 36
5.9 Part-of-speech tagging output. . . . . . . . . . . . . . . . . . . . . . . 37
5.10 Dependency annotated graph. . . . . . . . . . . . . . . . . . . . . . . 39
5.11 Dependency annotated graph. . . . . . . . . . . . . . . . . . . . . . . 40
5.12 Train data labelling using Prodigy. . . . . . . . . . . . . . . . . . . . 41
5.13 Named Entity Recognition graph. . . . . . . . . . . . . . . . . . . . 42
5.14 Information extraction graph. . . . . . . . . . . . . . . . . . . . . . . 44
5.15 Relation extracted. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
5.16 SPIED algorithm [140]. . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
5.17 Named Entity Recognition graph. . . . . . . . . . . . . . . . . . . . 46
6.1 Pipeline generated graph. . . . . . . . . . . . . . . . . . . . . . . . . . . 48
6.2 Manually generated graph. . . . . . . . . . . . . . . . . . . . . . . . . . 48
A.1 Part-of-speech and syntactic dependencies graph. . . . . . . 66
A.2 Dependency annotated graph. . . . . . . . . . . . . . . . . . . . . . . . 67
A.3 Named Entity Recognition graph. . . . . . . . . . . . . . . . . . . . . 68

List of Tables
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Data row example.</td>
<td>25</td>
</tr>
<tr>
<td>5.1</td>
<td>Syntaxnet Swedish language accuracy.</td>
<td>36</td>
</tr>
<tr>
<td>5.2</td>
<td>Main universal syntactic dependency relations [53].</td>
<td>38</td>
</tr>
<tr>
<td>5.3</td>
<td>Auxiliary universal syntactic dependency relations [53].</td>
<td>38</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

Traditionally, knowledge has been presented as something specific to the human being that is acquired or related to the "belief" in the existence of the rational soul that makes it possible to intuit reality as truth [1].

Knowledge was considered to respond to the intellective faculties of the soul [2] according to the three degrees of perfection of them: soul as the principle of life and vegetative self-movement, sensitive or animal soul and human or rational soul [3].

According to these postulates, all living beings acquire information from their environment through their faculties or functions of the soul [4]:

Vegetative
In vegetables to perform minimal vital functions innately, nutrition and growth, reproduction and death.

Sensitive
In the animals that produce adaptation and local self-movement and includes the former faculties. In the degree of superior perfection, memory, learning and experience appear, but in its degree, one can not reach the "true knowledge" of reality.

Rational
In the human being that, in addition to the previous functions, produces the knowledge by concepts that make possible the language and the conscience of the truth.

The merely material beings, inert, lifeless and soulless, have no knowledge or information about the environment, as entirely passive beings, only subject to material mechanical causality [3].

Experience, which is common to animals endowed with memory, does not yet offer a guarantee of truth because it is a subjective knowledge of those who have sensitive experience; which is valid only for those who experience it and only at the moment when they experience
1.1 Background

Nowadays most of the research efforts in the field of Computer Science pursue the dream of creating an Artificial Intelligence [8] that can be able to mimic or even replace human cognitive capacity. Unfortunately, the time has not yet arrived; challenges are yet to be tackled to achieve a fully capable system that resembles human brain processes [9].

One of the main challenges of this path to AI is the representation of knowledge and reasoning, an area of AI whose fundamental objective is to represent knowledge in a way that facilitates inference (draw conclusions) from that knowledge [10]. It analyses how to think formally or, in other words, how to use a system of symbols to represent a domain of discourse, together with functions that allow inferring (making formal reasoning) about objects.

An approach for solving the challenges of knowledge representation and reasoning are the Knowledge Bases and Knowledge-based systems [11], a technology used to store complex structured and unstructured information or data in conjunction with an inference engine that is able to reason about facts and use rules and other forms of logic to deduce new facts and query for specific information [12]. However, manually creating, maintaining and updating a knowledge base is a hard, time-consuming task [13], especially when the knowledge stored in it changes with high frequency. That is why automatic approaches using Machine Learning [14], and Natural Language Processing [15] are needed to achieve a solution to the problem of building systems for knowledge representation and reasoning as well as, at the same time, get closer to the dream of a fully functioning AI.
This work studies and implements a prototype of a system capable of extracting and represent knowledge in the format of knowledge graphs using ML tools and algorithms for NLP.

1.2 Problem

In this thesis the aim is to construct knowledge graphs, as a way of representing knowledge, extracting key terms and relationships from Swedish human-generated text.

Many other projects and research papers approach a solution to this problem with more or less success. For example, the work from Fraga, Vegetti et al. [16] Semi-Automated Ontology Generation Process from Industrial Product Data Standards and from Embley, Campbell et a. [17] Ontology-based extraction and structuring of information from data-rich unstructured documents where they build systems capable of generating a knowledge representation of domain-specific data. Also, the project from Al-Zubaide and Issa [18] OntBot: Ontology based ChatBot where they present an algorithm capable of extracting knowledge automatically from human conversation. However, none of them focus in other languages apart from English and also they are not very flexible regarding adding new features or steps to the process of extracting information.

To solve the problem, literature proved that ML could help with some of its tools and algorithms focused in enabling a machine to read and understand human language in a way that information can be extracted and structured for further processing or reasoning in this case [19].

The claim is that by building an NLP pipeline with custom steps and available technologies, a machine might be able to extract valuable information in the format of graphs that connect concepts, ideas and other information in a way that represents the knowledge being integrated in a given piece of human-generated text, not only for the Swedish language, but also others.

The research question that summarises the problem presented in this thesis report is:

*How can ML algorithms for NLP and Graph Databases be integrated and applied to build a Knowledge Graph using domain-specific human-generated Swedish text?*
1.3 Purpose

The purpose of this thesis is to present how different ML techniques in the field of NLP can be applied for extracting terms or concepts and relationships between them from Swedish, human-generated text and construct simple knowledge graphs that represent this knowledge.

1.4 Goal

The primary goal of this project is to implement a system capable of, using human-generated text in Swedish, extract the essential information from it and build and maintain a Knowledge Base with all of the data, not only provided by the input text but by further input that the system will eventually receive.

The input of the system may also be recorded voice or other formats of information which are subject to knowledge extraction.

Once the system is completed, it should be able to reason and provide answers to questions related to the domain of the information being stored in it.

1.4.1. Benefits, Ethics and Sustainability

Being able to automatically generate machine-readable knowledge representations have an immediate benefit of fostering the development of better information systems, as well as contributing to the creation of more advanced AI that will not only transform businesses and economy but also will have an impact on the society.

This kind of automatic information gathering systems have a possible impact on ethics since the data needed for training, testing and, in general, developing the system may contain private and sensitive information that needs to be handled with the highest security standards. But not only applying high-security standards will solve the ethics issue, but it is also imperative to adequately inform the user of what is going to be the use of its data and have the explicit consent to use it.

Finally, concerning sustainability, this project contributes explicitly to the ninth of the United Nations sustainable development goals [20] which relate to Industry, Innovation and Infrastructure. Specifically, this project is an investment in the field of information and communication technology, crucial for achieving sustainable development and empower communities in many countries. Also, according to the UN, technological progress is the foundation of efforts for achieving environmental objectives such as increased resource and energy-efficiency.
1.5 Research Methodology

There are two common methods for research to be categorised: *Quantitative research method* and *Qualitative research method* [21].

The *Quantitative Research* method uses mathematical, statistical or computational techniques to describe the causes of a phenomenon, relations between variables and to verify or falsify theories and hypothesis.

On the other hand, the *Qualitative Research* method refers to the meanings, concepts definitions, characteristics, metaphors, symbols, and description of things and not their numerical measures.

Since this thesis focuses primarily in developing a proof-of-concept of a pipeline that integrates a collection of ML algorithms, the method chosen is the *Qualitative* method, which focuses in the decisions taken for the implementation itself rather than their numerical values.

1.5.1. Philosophical Assumption

The philosophical assumption followed is the *Positivism*, which assumes that the reality is objectively given and independent of the observer and instruments [21]. This assumption may be the appropriate one since this thesis focus primarily on experimenting and testing different approaches that try to solve the problem stated.

1.5.2. Research Method

Concerning the Research method, this thesis is based on the *Applied* research, since it builds on existing research and uses real-world data for trying to achieve the goals described [21].

1.5.3. Research Approach

Finally, the research approach chosen is the *inductive*, which starts with the observations and theories are proposed towards the end of the research process as a result of observations [22]. The reason for choosing this approach relies on the fact that this research project begins with none strong hypotheses, the type or nature of the research findings will not be ready until the end of the study.
1.6 Delimitations

The primary focus of this work is to develop and test a system that solves the problem described in previous sections, as opposed to measuring the quality or performance of it at least on this first iteration of the project. Future work will be required for achieving a more performance-focused solution.

In the development of the proposed solution, risks such as time, data available, quality of that data and computing capacity have been taken into account. Likewise, measures have been adopted to limit its impact. Even so, until the moment of the presentation of the proposed project, there is no optimal solution for most of the NLP challenges within the field of ML. For this reason, it can not be guaranteed that the output of the system is entirely reliable or if the approach chosen for solving the problem is the most adequate.

1.7 Outline

This thesis report is structured as follows:

Chapter 2: Framework
The chapter two of this report presents to the reader an introduction to the framework of this thesis. Including a foreword for topics such as the Swedish language, Machine Learning, Natural Language Processing and so on.

Chapter 3: Related work
The chapter three of this report analyses the related work that focuses on common topics for this project.

Chapter 4: Methods
The chapter four of this report introduces the methods used for conducting the proposed thesis.

Chapter 5: The NLP Pipeline
The chapter five of this report presents the proposed solution to the problem stated, driving into the details of the chosen technology and the final implemented pipeline.

Chapter 6: Evaluation
The chapter six of this report provides the results of the evaluation of the system implemented concerning its accuracy.

Chapter 7: Conclusions and Future Work
The chapter seven of this report focuses on the conclusion and a self-evaluation of the thesis itself. It also introduces the future work that will be needed to improve and grow the project itself.
CHAPTER 2
Framework

This chapter presents the framework used to define a solution to the problem described. The first section will introduce to the reader the basics of Natural Language and its implications for this work.

After the concept of Natural Language, a comprehensive description of the main characteristics of Swedish language will be presented to provide a better insight into this language and its most relevant components. Finally, the concepts of Machine Learning, Natural Language Processing, its tools and Knowledge Representations will complete this chapter, citing the most relevant information needed for a better understanding of the project being gathered on this report.

2.1 Natural Language

The term natural language[23] designates a linguistic variety or form of human language with communicative aims that is endowed with syntax, and that obeys supposedly to the principles of economy and optimisation.

According to Hockett et al. [23] there are fifteen Natural Language features. To highlight some:

Mode of communication
Natural Language has two main channels as a mode of communication: vocal-auditory and manual-visual.

Broadcast transmission and directed reception
In a speech a message is broadcast that expands in all directions and can be heard by anyone; however, the human auditory system allows the identification of where it comes from.

Interchangeable development or interchangeability
A speaker, under normal conditions, can both broadcast and receive messages.
2.2 Swedish Language

Total feedback
The speaker can listen to himself at the precise moment he or she issues a message.

Semanticity
The signal corresponds to a particular meaning. It is a fundamental element of any method of communication.

Arbitrariness
There is no correlation between the signal and the sign.

Discreetity
The basic units are separable, without a gradual transition. A listener can hear either "t" or "d", and regardless of whether he hears it well, he will distinguish one or the other, without hearing a mixture of both.

Displacement
Reference can be made to situations or objects that are not situated by deixis, in the "here and now", that is, separated by time or distance, or even on things that do not exist or have not existed.

Double articulation or duality
There is a level or second articulation in which the elements have no meaning, but they distinguish meaning (phoneme), and another level or first articulation in which these elements are grouped to have meaning (morpheme). The elements of the second articulation are finite, but they can be grouped in infinite ways.

Productivity
The rules of grammar allow the creation of new sentences that have never been created but can be understood.

In a broader perspective, the concept of Natural Language can be summarised as a human communication procedure that does not need to follow a particular set of rules; instead, it evolves and changes with the time.

2.2 Swedish Language

Swedish is a Scandinavian language [24] currently being spoken by more than nine million people in the world, mainly in Sweden and parts of Finland, especially on the coast and in the Åland islands [25]. Like other Scandinavian languages, Swedish descends from the Old Nordic, the common language of the Germans who remained in Scandinavia during the Viking era [15].

Standard Swedish is the language that evolved from the dialects of Central Swedish in the 19th century and was already well established
2.2 Swedish Language

in the early twentieth century [26]. Although there are still different regional varieties that come from ancient rural dialects, the spoken and written language is uniform and standardised. Some dialects are very different from standard Swedish in grammar and vocabulary and are not always mutually intelligible with standard Swedish [27]. These dialects are limited to rural areas and are mainly spoken by a small number of people with little social mobility. Although they are not threatened with imminent extinction, these dialects have been declining during the last century, although they are well documented, and their use is encouraged by local authorities [28].

In Standard Swedish, the standard order of the phrases is Subject - Verb - Object, although this can often change to emphasise certain words or phrases. The morphology of Swedish is similar to English, that is, it has relatively few flexions; two genres, no case, and a distinction between singular and plural. Adjectives are compared as in English, and they are also flexed according to gender, number and effect of definition. The effect of defining the nouns is mainly marked by suffixes (terminations), complemented with definite and indefinite articles. Prosody presents both accent and tones. The language has a relatively large vowel variety.

2.2.1. Alphabet

The Swedish alphabet is a variant of the Latin alphabet [29]. It consists of 29 letters, of which three (å, ä and ö) are characteristic of this language and other Germanic languages. The Swedish vowels are a, e, i, o, u, y, å, ä and ö. It contains the 26 letters of the Latin alphabet.


2.2.2. Nouns

A noun in Swedish is, as in other Indo-European languages, a class of words meaning things, objects, places, people, plants, animals, phenomena and abstract concepts. It is a different part of speech: it takes a definite or undefined form, it can have a singular and plural number, rudimentary forms of complement cases, and each gender is assigned a type: utrum or neuter. In the sentence, it can act as a subject, object or attachment [24].

2.2.3. Adjectives

Swedish adjectives are declined according to gender, number, and definiteness of the noun [24].
2.2.4. Adverbs

The adverb is an invariable part of speech in the Swedish language, functioning in the sentence as an observer. A large part of Swedish adverbs is created by adding the adjective tip t to the basic form.

Some adjectives ending in -lig form adverbs by adding the ending -en or -vis. Adverbs ending in -vis and -en often appear as conditions for circumstance or consent.

Other endings of adverbs are: -ledes / -lunda. Many adverbs function completely independently, without being a derivative of any other expression [24].

2.2.5. Numbers

In Swedish, we distinguish between primary and order numerals. They may in the sentence act as the subject, object and attachment [24]. Natural numbers from 0 to 10: noll, en, ett, två, tre, fyra, fem, sex, sjue, åtta, nio, tio.

Creating the second ten numbers (between 11 and 20) is characterised by high irregularity. The -ton ending is accompanied by numbers from 13 to 20, which is the effect of the numeral system in the period of using the duodecimal system. Only three numerals: 15, 16 and 17 are created entirely regularly.

Unlike other Scandinavian languages, numbers between 21-99 are arrayed by the number of units: tjugo, tjugoen, tjugotvå. The end of numbers from 30 to 90 is -tio.

2.2.6. Pronouns

In Swedish, depending on the replaced part of the speech pronouns are divided into noun, adjectival, adverbial and numeral. Due to the syntactic functions, noun pronouns are called independent, and adjective pronouns [24].

Some pronouns can have both noun and adjective functions so that they can be used as a substitute for a noun or as a term. The word to which the pronoun refers or which it replaces is called the correlate of the pronoun. From the degree of specificity of the correlate, we divide the pronouns in Swedish into three groups [24]:

- **Definite pronouns**: Which replace a strictly defined correlate and includes:
  - Personal pronouns
  - Possessive pronouns
  - Demonstrative pronouns
2.2 Swedish Language

- Reflexive pronouns
- Pronounced pronouns
- Querying pronouns

- **Query pronouns**: Which express the question about correlates.
- **Indefinite pronouns**: Whose correlates are not specified.

2.2.7. Prepositions

The preposition is an invariable part of speech expressing the relation between particular elements of a sentence or whole sentences. It is mostly unstressed. The use of a specific preposition does not change the form of the word. After the preposition, there is usually a pronoun in the form of a complement. Many verbs have predetermined prepositions [24]. Prepositions can be:

- Simple
- Complex
- Complete
- Derivatives

2.2.8. Verbs

The verb in Swedish is a different part of speech, although the paradigm of the variety is simplified: its form depends neither on the person nor the number. The starting form of the verb to create other forms is the imperative mode, which is also the shortest existing form of the verb. Thus, the verb adopts, basically, the following morphological forms [24]:

- Infinitive
- Present tense
- Past tense, imperfect
- Past tense, perfect
- Future tense
- Supine
2.2.9. Conjunctions

The conjunction is an unchangeable part of the sentence, and its task is to combine individual words or entire sentences in compound sentences. Conjunctions concerning their function are divided into coordinates and subordinates [24].

2.2.10. Syntax

Typically, the Swedish language belongs to the positional language group like German or English [30]. The syntax is an important element of Swedish grammar. Swedish is a Germanic language, Swedish syntax shows similarities to both English and German. It has a Subject - Verb - Object word order. Like German, Swedish utilises verb-second word order in main clauses, for instance after adverbs, adverbal phrases, and dependent clauses. Adjectives generally precede the noun they determine [24].

![Diagram of Swedish sentence structure analysis](image)

*Figure 2.1: Sample Swedish sentence structure analysis [31]*

2.3 Machine Learning

Machine Learning is currently one of the most outstanding subfields of Computer Science [32]. It is a branch of the Artificial Intelligence world, and its fundamental aspiration is to develop techniques that allow computers to learn [33].

As a matter of fact, Machine Learning is about creating applications capable of generalising behaviours from information provided in the form of examples. It is, therefore, a process of knowledge induction. In many cases, the field of action of Machine Learning overlaps that of computational statistics [34], since the two disciplines are based on data analysis [35]. However, machine learning also focuses on the study of the computational complexity of problems.
The process of learning can be divided into two main groups, which also serve as the two major groups for classifying Machine Learning techniques:

- Supervised learning
- Unsupervised learning.

Supervised learning is a technique to deduce a function from training data [36]. The training data consist of pairs of objects (usually vectors): the first component is the input data and the other, the expected result. The output of the function can be a numeric value (as in regression problems) or a class label (as in classification). The target of this type of learning is to create a function capable of predicting the output corresponding to any valid input object after having seen a series of examples (or training data). For this, it has to generalise from the data presented to the situations not previously seen.

In unsupervised learning, on the contrary, there is no a priori knowledge [37]. Thus, unsupervised learning typically treats input objects as a set of random variables, with a density model being constructed for the data set [38].

### 2.4 Natural Language Processing

Natural Language Processing can be defined as a collection of Machine Learning techniques and algorithms that enables computers to analyse, understand, and derive meaning from human language [39]. NLP, in contrast to standard text processors, considers the hierarchical structure of language: several words make a phrase, several phrases make a sentence and, ultimately, sentences convey ideas [40].

NLP can also organise and structure knowledge, to perform tasks such as automatic summarising [41], translation [42], named entity recognition [43], relationship extraction [44], sentiment analysis [45], speech recognition [46] and so on.

In general, processes involving Natural Language Processing are composed by a sequence of tasks [47]. Each of those tasks has a specific purpose, such as tokenising text to analyse its words, label each of those words with their respective grammar category and so on. The next subsections will describe each of those Natural Language Processing tasks that shape a typical pipeline.

#### 2.4.1 Word and sentence tokenisation

Detecting token and sentence boundaries is an important preprocessing step in Natural Language Processing applications since most of these
operate either on the level of words or sentences. The primary challenges of the tokenisation task stem from the ambiguity of individual characters in alphabetic and the absence of explicitly word boundaries in most of the languages [48].

The process of finding boundaries of sentences in a text is often done heuristically, using regular expressions. Although best performance is achieved with supervised Machine Learning models that predict, for example, for each full stop whether it denotes the end of the sentence [49].

### 2.4.2. Part-Of-Speech tagging

In computational linguistics, grammar labelling (also known as, part-of-speech tagging) is the process of assigning to each of the words in text its grammatical category [50]. This process can be done according to the definition of the word or the context in which it appears, for example, its relation to adjacent words in a sentence or a paragraph.

The grammar labelling is more complicated than it seems at first glance, it is not as simple as having a list of words and their corresponding grammatical categories since some words can have different grammatical categories depending on the context in which they appear [51]. This fact often occurs in natural language where a large number of words are ambiguous. For example, the word ‘given’ can be a singular name or a form of the verb ‘give’. That is why Machine Learning algorithms for Natural Language Processing are very useful when trying to tackle down this problem.

There is also another challenge related to the labelling format used. Since every language in the world is unique, it is hard to agree on one way of labelling the same type of words [52]. That is why the project Universal Dependencies has been making a great effort to try to coordinate the work being done around the world related to grammar labelling. To solve that, Universal Dependencies is a project that is developing cross-linguistically consistent treebank annotation for most of the world’s languages, with the goal of facilitating multilingual parser development, cross-lingual learning, and parsing research from a language typology perspective [53]. The annotation scheme is based on an evolution of Stanford dependencies [54], Google universal part-of-speech tags [55], and the Interset interlingua for morphosyntactic tagsets [56].

### 2.4.3. Syntactic Dependency Parsing

Syntactic Dependency Parsing or Dependency Parsing consists on assigning a syntactic structure to a given well-formed sentence. It is based on the dependency grammar theory [57], which its fundamental basis
is to consider not only the lexical elements that make up the syntactic structure but also the relationships between them [58]. According to this theory, there is a link that relates to the ideas expressed by the words to form an organised thought [59]. This link is called a connection.

The connections operate hierarchically; thus, when joining two elements, one will be considered as regent and the other as governed. When a ruler has several regents a knot is built, which are staggered in a complex system of dependencies, which culminates in the central knot.

![Figure 2.2: Grammatical Dependency Graph.](image)

The syntactic dependency parsing analysis represented by a diagram, or a dependency graph as shown in Figure 2.2.

### 2.4.4. Named Entity Recognition

The recognition of named entities is one of the most important topics in Natural Language Processing [60]. It is a subtask of information extraction [61] that aims to provide a predefined label or class to entities in a given unstructured text.

In the *named entity* expression, the word *named* restricts the task to those entities for which there is a common designator or label. For example, in the sentence: *Barack Obama was born in 1961 in Honolulu, Hawaii* a possible named entity recognition system may produce:

- **Barack Obama**: PERSON
- **1961**: DATE
- **Honolulu**: PLACE

Recognition of named entities is often divided into two distinct problems [62]: name detection, and name classification according to the type of entity they refer to.
The first phase is usually reduced to a segmentation problem [63]: the names are a contiguous sequence of tokens, without overlapping or nesting, so that Barack Obama is a unique name, although that within this name appears, the substring Obama which is itself a name.

Temporal expressions and some numerical expressions (money, percentages and so on) can also be considered entities in the context of the Named Entity Recognition. While some cases of these types are good examples of rigid designators, there are also many invalids [64], for example, Barack Obama was born in August. In the first case, the year 1961 refers to the 1961 year of the Gregorian calendar. In the second case, the month of August can refer to the month of any year. The definition of a named entity is not strict and often has to be explained in the context in which it is being used.

2.4.5. Relation and Information Extraction

Relationships, in the context of information retrieval [65], are the grammatical and semantic connections between two entities in a sentence.

The task of extracting information from unstructured text aims to automatically extract structured, or semi-structured information from documents readable by a computer [66] is also one of the most significant challenges in the context of Natural Language Processing [67].

![Figure 2.3: Information Extraction example. Source: nlp.stanford.edu](image)

In the past few decades, different approaches to solve this problem have been studied and implemented with more or less success [68], most of the time involving the use of domain ontologies [69] which proved to not be cost-efficient and difficult to maintain [70]. Other approaches involved some new techniques such as Machine Learning [71] which proved to leverage most of the previously implemented solutions [72].
2.5 NLP tools and frameworks

The current state of the art of the field of Natural Language Processing involves new tools and frameworks that serve as building blocks for all the projects that require, in all levels of complexity, to process Natural Language.

This section presents the four major projects that currently are the most widely used, accepted and validated for such NLP tasks [36] [73].

<table>
<thead>
<tr>
<th>Comparison</th>
<th>tokens for manual annotation</th>
<th>tokens for library</th>
<th>identical tokens</th>
<th>identical tokens (%)</th>
<th>identical tokens(POI)</th>
<th>identical tokens(POI) (%)</th>
<th>identical tokens(POI)</th>
<th>identical tokens(POI) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual vs. Stanford</td>
<td>375</td>
<td>385</td>
<td>371</td>
<td>97.43</td>
<td>339</td>
<td>89.21</td>
<td>317</td>
<td>83.42</td>
</tr>
<tr>
<td>Manual vs. SyntaxNet</td>
<td>375</td>
<td>366</td>
<td>357</td>
<td>96.36</td>
<td>332</td>
<td>89.61</td>
<td>317</td>
<td>85.56</td>
</tr>
<tr>
<td>Manual vs. spCy</td>
<td>375</td>
<td>377</td>
<td>369</td>
<td>98.14</td>
<td>347</td>
<td>82.29</td>
<td>328</td>
<td>89.89</td>
</tr>
<tr>
<td>Manual vs. NUK</td>
<td>375</td>
<td>374</td>
<td>375</td>
<td>99.60</td>
<td>331</td>
<td>88.38</td>
<td>306</td>
<td>81.71</td>
</tr>
</tbody>
</table>

![Figure 2.4: NLP tools and frameworks accuracy evaluation [73].](image)

2.5.1. Stanford CoreNLP

Stanford CoreNLP provides a set of human language technology tools [74]. It is an integrated framework which aims to facilitate the application of linguistic analysis to human-generated raw text.

It is currently the most commonly used library in software engineering research related to the processing of Natural Language in all its levels since it has a wide range of grammatical analysis tools and support for all the major languages.

Amongst its features, the most interesting ones are: giving the base forms of words, their parts of speech, Named Entity Recognition, syntactic dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract particular or open-class relations between entity mentions, and so on.

2.5.2. NLTK Python Library

NLTK is a platform for building Python programs to work with human-generated text data. It incorporates a suite of text processing libraries for classification, tokenisation, stemming, tagging, parsing, and semantic reasoning and also wrappers for the leading industrial-strength NLP libraries [75]. Similar to the Stanford CoreNLP library, NLTK provides many integrations for different programming languages and comes with many resources.
2.6 Knowledge Representation

2.5.3. spaCy

SpaCy[76] is a library also written in Python [77] and Cython [78]. It is primarily aimed at commercial applications since it is substantially faster than many other libraries. It includes all the features of the previously introduced libraries.

2.5.4. SyntaxNet

SyntaxNet[79] is a framework for syntactic parsing built as a Neural Network [80] implemented in Google’s Deep Learning framework TensorFlow [81]. Given a sentence as input, it tags each word with a part-of-speech tag that describes the word’s function in a given sentence, and it determines the relationships between those words, represented in the dependency parse tree. It is currently one of the most accurate parsers, achieving, on average and depending on the language, more than 90% of accuracy.

2.6 Knowledge Representation

Knowledge representation and reasoning is a field of Artificial Intelligence whose fundamental purpose is to represent knowledge in a way that facilitates inference, that is, drawing conclusions from that knowledge [82].

There is a set of representation techniques [83] such as frames, rules, labelling and semantic networks, which have their origin in theories of human information processing. As knowledge is used to achieve intelligent behaviour, the fundamental objective of knowledge representation is to represent knowledge in a way that facilitates reasoning [84]. A good representation of knowledge must be declarative, in addition to fundamental knowledge.

In general, knowledge representation can be understood concerning five fundamental roles [85]:

- A representation of knowledge is fundamentally a substitute, a substitute for an object from real life.
- It is a group of ontological commitments, an answer to the question about the terms in which one should think about the world.
- It is a part of the theory of intelligent reasoning, expressed regarding three components: The fundamental concept of the representation of intelligent reasoning; The set of inferences that the representation sanctions; The set of inferences that it recommends.
It is the computational environment in which thought takes place, given by the guidance that a representation provides a method to organise information in a way that facilitates making inferences.

It is a mode of human expression, a language in which things about the world are expressed.

In the field of artificial intelligence, problem-solving can be simplified with an appropriate choice of knowledge representation [86]. Some problems are easier to solve by representing knowledge in a certain way.

### 2.7 Graph databases and Knowledge graphs

A graph-oriented database [87] represents the information as nodes of a graph and its relationships with the edges of it, so graph theory [88] can be used to traverse the database since it can describe attributes of the graphs nodes (entities) and the edges (relationships) [88].

![Graph database example](image)

*Figure 2.5: Graph database example.*

A graph database must be standardised; this means that each table would have only one column and each relation only two, with this being achieved that any change in the structure of the information has only a local effect [89].

The graph databases also offers new or improved characteristics such as broader queries and not demarcated by tables [90]. Also, there is no need to define a certain number of attributes. Moreover, the records are also of variable length, avoiding having to define a size and also possible faults in the database. Besides, graph databases can directly be navigated hierarchically, getting the root node from the leaf node and vice versa.

In the field of Knowledge Representation, Graph databases present a crucial role [86], since its main advantage is the versatility, as they can store both relational and complex semantic data [91]. The example this conjunction between graph databases and knowledge are the Knowledge Graphs [92], a representation of knowledge in the format of graphs, which offer a more powerful way of representing knowledge.
Figure 2.6: Knowledge graph example.
CHAPTER 3
Related Work

This chapter introduces the most relevant related work for the topic being considered in this project. First and foremost, it will focus on the areas of Natural Language Processing systems, techniques and frameworks for information and knowledge extraction for building knowledge graphs. Besides, modern knowledge representation techniques and research will be introduced. Finally, it will conclude with an introduction to relevant research and projects for Machine Learning, and Natural Language Processing approaches.

3.1 Information and Knowledge extraction

In the area of information and knowledge extraction it is important to highlight the work of Jingbo, Jialu et al. Automated Phrase Mining from Massive Text Corpora [93] which presents an interesting approach to the task of automated phrase mining, which leverages a large amount of high-quality phrases in an effective way, achieving better performance compared to limited human labeled phrases. Parts of this strategy will be used for sentence tokenisation and phrase from text extraction during the implementation of the proposed solution in further chapters.

On the other hand, it is also interesting the survey done by Krzywicki, Wobcke et al. Data mining for building knowledge bases: Techniques, architectures and applications [61] which analyses the state-of-the-art literature on both current text mining methods (emphasising stream mining) and techniques for the construction and maintenance of knowledge bases. Specifically, its focus on mining entities and relations from an unstructured human-generated text that presents a challenge concerning entity disambiguation and entity linking. This survey is the main inspiration for the steps taking in the solution to extract entities and its relations.

Also interesting is the review on relation extraction done by Bach and Badaskar A review of relation extraction [68] which focus on methods
for recognising relations between entities in unstructured text. Also part of the inspiration on the information retrieval parts of this project.

Finally, it is also interesting to cite the following research projects: Leveraging Linguistic Structure For Open-Domain Information Extraction [94], A text processing pipeline to extract recommendations from radiology reports [95], Fast rule mining in ontological knowledge bases with AMIE+ [96] and Semi-Automated Ontology Generation Process from Industrial Product Data Standards [16] which also influenced to a greater or lesser extent the decisions taking for solving the problem proposed.

3.2 Knowledge representation

Another important topic related to the project being discussed in this report is the approaches for effectively representing knowledge.

In this area, it is essential to mention the paper from Pilato, Augello et al. A Modular System Oriented to the Design of Versatile Knowledge Bases for Chatbots [97] which illustrates a system that implements a framework oriented to the development of modular knowledge bases and also part of the justification in terms of the decisions taken in order to represent knowledge in this project.

To add on the field of knowledge representation the work from Schuhmacher and Ponzetto Knowledge-based graph document modelling [98] which propose a graph-based semantic model for representing document content also motivated the decisions on how to build the knowledge graphs.

Finally, more on this topic can be found in the following papers: WHIRL: a world-based information representation language [99], ScaLeKB: scalable learning and inference over large knowledge bases [100], Experience-based knowledge representation: Soeks [101], Ontology-based extraction and structuring of information from data-rich unstructured documents [102] and Open-domain Factoid Question Answering via Knowledge Graph Search [103].

3.3 Machine Learning & Natural Language Processing

One of the most crucial components of the presented work is the Machine Learning techniques for Natural Language Processing.

One of the most interesting readings is the paper presented by Lavelli, Calif et al. Evaluation of machine learning-based information extraction algorithms: Criticisms and recommendations [72] which surveys the evaluation methodology adopted in information extraction, as defined in a few different efforts applying machine learning to Information Ex-
traction and served as a methodology for evaluating the system implemented.

Moreover, the research from Ireson, Califf et al. *Evaluating machine learning for information extraction* [71] which also presented an interesting approach for evaluating Machine Learning techniques for Information Extraction.

Finally, also important to mention the next two papers: *A review of relational machine learning for knowledge graphs* [104] and *R-Net: Machine Reading Comprehension With Self-Matching Networks* [105].
CHAPTER 4
Methods

This chapter introduces the methods used for both obtaining the data required for the system and the software development techniques used for implementing the system.

4.1 Data collection

Data collection is the process of collecting and measuring information on variables of interest, in a systematic methodology established to allow someone to answer questions from presented research, test hypotheses and evaluate results [106]. The research component of the data collection is common to all fields of study, including the physical, social, human and business. While methods vary by discipline, the emphasis on certifying an accurate and honest collection remains the same [107].

Regardless of the field of study or preference of data definition (quantitative or qualitative), accurate data collection is essential to maintain the integrity of research [108]. Both instrument selections appropriate for the collection (existing, modified, or newly developed) and deliberately clear instructions for its correct use in order to reduce the likelihood of errors occurring [109].

A formal data collection process is necessary to ensure that the data obtained are as well defined as accurate and that subsequent decisions on arguments based on the information found are valid. The process creates a baseline in which to measure and in some instances a target to be improved [110].

Concerning this research project, data collected is Qualitative [111], since it consists of extraction from the support forum of Telia Company website\(^1\). This data contains threads of conversation between customers and support agents. The format of the data extracted is Comma Separated Values which contains information such as conver-

\(^1\)https://forum.telia.se
4.2 Data analysis

Qualitative and Quantitative data also require different types of analysis to be applied. In the case of qualitative-oriented researches, where the data falls into the types of interviews, text or other non-numerical data analysis involves identifying common patterns within the responses and critically analysing them in order to achieve research aims and objectives [114]. Data analysis for quantitative data [115], on the contrary, involves analysis and interpretation of numerical data and attempts to find the rationale behind findings [116].

As previously discussed, the data gathered for this project is, in general, qualitative, which refers to non-numeric information such as text documents. That means a qualitative data analysis will be applied.

Qualitative data analysis can be divided into the following categories [117]:

- **Content analysis** Which is the process of categorising verbal or behavioural data.

- **Narrative analysis** Which involves the reformulation of stories presented by respondents of a survey taking into account the context.

- **Discourse analysis** Which systematically studies written and spoken discourse as a form of the use of language.
4.3 Data verification and validation

- **Framework analysis** Which consists of several stages such as familiarisation, identifying a thematic framework, coding, charting, mapping and interpretation.

- **Grounded theory** It begins with an analysis of a single case to formulate a theory. Later on, additional cases are examined to see if they contribute to the theory established at the beginning.

For the project being discussed in this report, the principal data analysis conducted with the data collected from Telia’s forum is a **Discourse analysis** [118], focusing primary on the text grammar, discerning about well-formed sentences (according to the grammar) and non-well formed ones, as it is shown in Figure 4.1. The rationale behind this decision focuses primarily on the fact that for the required data that will be fed to the implemented solution, it is essential to have available quality text where the sentences are well formulated, and its grammar is correct.

![Grammatically correct sentences](image)

*Figure 4.1: Grammar analysis.*

### 4.3 Data verification and validation

The processes of verification and validation of the quality or reliability of the information obtained through qualitative methods are different from those obtained through quantitative methods [119]. The verification of the reliability of qualitative information is an essential element of the design and development of the study that increases the quality of the information collected [120]. It is not a group of tests applied to the collected data (unlike the statistical tests), but instead of verifications made before starting the data collection and monitored during the research [121].

Regarding the verification of qualitative data strategies, it can be distinguished between [122]:

- **Grounded theory** It begins with an analysis of a single case to formulate a theory. Later on, additional cases are examined to see if they contribute to the theory established at the beginning.

For the project being discussed in this report, the principal data analysis conducted with the data collected from Telia’s forum is a **Discourse analysis** [118], focusing primary on the text grammar, discerning about well-formed sentences (according to the grammar) and non-well formed ones, as it is shown in Figure 4.1. The rationale behind this decision focuses primarily on the fact that for the required data that will be fed to the implemented solution, it is essential to have available quality text where the sentences are well formulated, and its grammar is correct.
4.4 Software development

- **Methodological coherence** Which ensures consistency between the research question and the components of the method. The development of the research is not a linear process since the qualitative paradigm allows flexibility to adapt the methods to the data throughout the process of collecting them.

- **Proper sampling** Which consists in recovering the participants that best represent or have knowledge about the subject in question, thus achieving an efficient and efficient saturation of the categories, as well as replication of patterns.

- **Collection and analysis of concurrent information** Which form a mutual interaction between what is known and what one has to know. This dynamic and flexible interaction is requested by the data itself, thus changing, if necessary, the preliminary design chosen by the researcher.

- **Theoretical thinking** The ideas coming from the data are reconfigured in new data, which give rise to new ideas that must, in turn, be contrasted with the data that is being collected.

- **Development of theory** Which means to move with deliberation between the micro perspective of the data and a macro conceptual/theoretical understanding.

During the development of this project, data collected has been monitored through the process of extracting it until the usage to guarantee compliance with the strategies mentioned before.

### 4.4 Software development

The software development process [123](#) (or software development life cycle) is a part of the software development which is defined as a sequence of activities that must be followed by a team of developers to generate or maintain a coherent set of developed software or products. In this context, a process can be defined as a framework of work that helps the control regarding management and engineering activities. This helps establish the "who does what, when and how" and a control mechanism to see the evolution of the project.

The Rational Unified Process (RUP)[124](#) is a software development process proposed by the Rational Software company which incorporates knowledge called best practices and is supported by a set of tools, such as Unified UML Modelling Language [125](#). These methods offer modelling techniques for the different stages of the development process, including individual annotations, graphs and criteria for developing quality software.
4.4.1. Software development stages

The most common stages of a typical software development process are [126]:

Requirements analysis
The main objective of this stage is to obtain the document specifying the requirements [127]. This document is known as a functional specification. The specification is understood as the task of writing in detail the software to be developed in a mathematically rigorous way. These specifications are agreed with stakeholders, which usually have a very abstract view of the final result, but not about the functions that the software should fulfil. The analysis of requirements is regulated by the IEEE SA 830-1998 [128] document which contains recommended practices for the specification of the requirements of the software.

Design and architecture
This stage determines the general functioning of the software without entering specific details about its technical implementation. The necessary technological implementations, such as the hardware and the network, are also taken into account.

Programming
The complexity and duration of this stage depend on the programming languages used, as well as the previous design generated.

Tests
At this stage, software developers verify that it worked as expected and specified in previous stages. This is a crucial stage because it allows detecting errors in the operation of the software before being delivered to the end user. It is considered a good practice that tests are carried out by people different from software developers.

Deployment and maintenance
The deployment consists of the delivery or implementation of the software to the end user. It occurs once it has been duly tested and evaluated.

4.4.2. Software development models

Software development models are an abstract representation of the stages defined in the previous subsection [126]. They can be adapted and modified according to the needs of the software during the development process.

The number of available software development methods is vast [129]. Amongst others, the most common ones are:
Cascade model
Also known as a traditional model, it is called "cascaded" by the position of the development phases in it, which make the impression that they fall as if they were a waterfall (down) [126]. In this case, the stages have a course strictly, that is, until a phase has not finished, it does not "go down" to the next. Reviews can be done before beginning the next phase, which opens up the possibility of changes, as well as ensuring that the phase has been executed correctly.

Prototype model
The software prototype model consists in the construction of applications (incomplete prototypes) that show the functionality of the product while it is in development, given that the final result may not follow the logic of the original software [126]. It enables the stakeholders to know the software requirements in the early stages of the development process. It also helps developers understand that the product is expected to be in the development phase.

Incremental development model
The incremental development model consists of increasing the content of the phase-by-stage model (design, implementation and testing phase) constantly until the software is finished. It includes both the development of the product and its subsequent maintenance. The main advantages are that after each iteration a review can be made to verify that everything has been corrected and the errors that may have been found can be changed [126]. In this case, the stakeholders can respond to the changes and constantly analyse the product for possible changes.

Iterative and incremental development model
The model obtained by combining the iterative model with the incremental development model [126]. The relationship between them is determined by the methodology of development and the process of building the software. It facilitates the process of detecting important errors and problems before they are manifest and can cause a disaster.

Spiral model
The activities of this model are formed in a spiral, in which each loop or iteration represents a set of activities. These activities are selected according to the risk factor they represent, starting with the previous loop [126]. It combines key aspects of the cascade model and the fast application model to try to take advantage of both models.

Rapid application development
The Rapid Application Development (RAD) model combines the
iterative development with the rapid construction of prototypes instead of long-term planning [126]. The lack of this planning usually allows the software to be written much faster and makes the change of requirements easier. The rapid development process begins with the development of preliminary data models and the use of structured techniques for processing models. In the next phase, the requirements are verified through prototypes. These phases are repeated iteratively.

**Agile development model**

It is a group of software development methodologies based on the iterative development, where requirements and solutions evolve through collaborations between organised teams [126]. Iterative development is used as a basis to advocate a lighter point of view and centred on people than in the case of traditional solutions. Agile processes use feedback instead of planning as the main control mechanism. This feedback is channelled through periodic tests and various versions of the software.

In the case of this thesis, the software development methodology used is based on the Agile development model [130], based on the iterative development, where requirements and solutions evolve through collaborations between organised teams. Iterative development is used as a basis to advocate a lighter point of view and centred on people than in the case of traditional solutions. Agile processes use feedback instead of planning as the central control mechanism. This feedback is channelled through periodic tests and numerous versions of the software.
CHAPTER 5

The NLP Pipeline

This chapter presents a description of the solution proposed and the system implemented. It also provides a comprehensive explanation of the decisions taken concerning the design, technologies, strategies and approaches used to develop the system.

Chapter’s design section proposes a layout strategy to take up the problem stated at the introduction of this document. Further sections within this chapter will exhibit the actual implementation, referencing frameworks, tools and technologies used in each part of the system.

To clarify, provide context and exemplify each part of the implemented pipeline, through this chapter a sentence will be utilised as an example input data and to explain how it will traverse each of the steps, ending up building the actual knowledge graph. This mentioned sentence is När den inkluderade surfmängden i ditt abonnemang är slut kan du fylla på med mer för att fortsätta att surfa på MittTelia and its translation to English: “When the included amount of surfing in your subscription is over, you can fill in more to continue browsing at MittTelia”.

5.1 Design

The underlying idea behind the goal of this project is to produce simple knowledge graphs that represent the information enclosed in any given well-formed (conforms to the grammar of the language of which it is a part) phrase or batch of text in the Swedish language. Since this goal is broad, during this design phase, it will be narrowed to smaller and simpler tasks or processes.

Generate Information triples  
Integrate Information triples  
Construct knowledge graph

Figure 5.1: Main tasks identified.
After studying the problem statement, three main tasks have been identified. As the Figure 5.1 shows, those tasks are:

- **Generate information triples**: Extract the basic information units from a given text in the format of \((e1, r, e2)\) triples, where \(e1, e2\) are entities or concepts and \(r\) is the relation that links those two concepts.

- **Integrate information triples**: Once the triples are generated, integrate them so all the important information within the input text data is gathered.

- **Construct knowledge graph**: Finally, construct the knowledge graph representing the information contained in any given text.

In order to represent this idea, given an example sentence *Telia, ett telekomföretag, är verksamt i Sverige* the next two Figures show how the extraction of triples is made.

![Figure 5.2: Sample triple extracted.](image1)

![Figure 5.3: Sample triple extracted.](image2)

As shown in Figures 5.2 and 5.3 the phrase: *Telia, ett telekomföretag, är verksamt i Sverige.* has been transformed into a triple that represents part of the information being disclosed in the utterance.

Following the task flow described above, the Figure 5.4 will represent the final knowledge graph extracted from the given sentence.

As described along this section, the proposed solution needs to solve the three main tasks proposed. To do so, specific tools, technologies and frameworks are to be used and integrated to produce the required output. The next sections will explain, in more detail, how the main tasks defined are implemented.
5.2 Generating information triples

The first task is to generate information triples with presented in Figure 5.5. Those triples represent fundamental information in the given input text data. Those triples are composed of two entities (concept, idea or action) and one relation (that establishes the primary relationship).

The task of generating those triples is not trivial and requires different sub-tasks to accomplish its purpose. The following subsections describe the process followed to transform a chunk of text into a straight information triple.

5.2.1 Text preprocessing

The primary source of input data for this project is human-generated text from different sources such as manuals, social networks, internal databases and forums. Therefore it is expected that the data would require to go into a cleaning process to make it suitable for further processing.

According to the Cross-Industry Standard Process for Data Mining [131], preprocessing and data cleansing are essential tasks that generally must be carried out so that the data can be used effectively for tasks such as Machine Learning. Raw data (such as the one extracted from the sources stated above) is often noisy and unreliable, so the use of this data for modelling can produce misleading results. These tasks typically follow an initial exploration of the data that is going to be used to detect and plan the necessary preprocessing.
Preprocessing and cleaning tasks can be performed in a wide variety of environments and with various tools and languages, such as R [133] or Python [77], depending on where the data is stored and how it is formatted. This section presents several concepts of data processing and tasks that will be carried out before entering data in the system.

Text cleaning

The first step is cleaning the text of unwanted and unnecessary characters or words. This includes: escaping HTML characters, removing punctuation signs, URLs, unrecognised characters for the given language and emojis or other types of unwanted encoded text.

To perform the mentioned text cleaning tasks, the Python [134] library NLTK (Natural Language Toolkit) [75] will be used. This tool is especially interesting since include functions such as:

- Text tokenisation
- Punctuation removal
- Case normaliser
- Stop words filtering
- Word stemming
- Text cleaning using regular expressions

The decision regarding which of this functions are going to be used will depend on a previous observation and analysis of the data gathered. Hence, not all the functions are always required.
5.2 Generating information triples

Word and sentence tokenisation

Once the data, in this context: text, has been cleaned, the next step is to tokenise it. Grefenstette et al. quoted: "Any linguistic treatment of freely occurring text must provide an answer to what is considered a token." [135] Which proves the importance of having a tool or method capable of splitting text into sentences and tokens in a meaningful way.

![Diagram](image)

**Figure 5.7:** Text transformations before linguistic analysis [135].

For this project scope, a token can be understood in two different ways or levels:

- Each of the words that compose a sentence.
- Each of the sentences that compose a chunk of text.

This step will specially focus on the second level of tokenisation, which is meant to recognise sentence boundaries since it is the unit of the system being implemented consider as the unit of treatment. But not to be forgotten that word level tokenisation is also needed for further processing steps that will be described later.

The tool chosen to perform this task is the Syntaxnet [136] project inside TensorFlow [81] framework for large-scale machine learning created by Google. Syntaxnet provides a specific Machine Learning model for word and sentence tokenisation that has been trained specially for the Swedish language1.

The output of this step is a collection of adjacency graphs that represent each of the detected sentences. Those graphs are composed of linked work tokens that follow a relationship described by the word next. This simple relationship indicates which word follows each other in a given sentence, as the Figure 5.8 shows. The reason behind keeping the word order is to conduct metrics and statistics around word order, but lacks interest within the goal of this project, since that is part of the future work.

---

1Syntaxnet Swedish model: [https://github.com/tensorflow/models](https://github.com/tensorflow/models)
5.2 Generating information triples

Figure 5.8: Text adjacency graph example.

5.2.2 Part-Of-Speech tagging

As introduced before, part-of-speech tagging is the process of classifying words into their parts of speech and labelling them accordingly. The Syntaxnet framework also offers a Swedish part-of-speech tagger that uses the Universal Dependencies, introduced previously on this document, to perform a grammar labelling for Swedish text. As we can see in the Figure 5.1 the accuracy of this tagger (in average) is above 90% which means that for most of the times the POS label representing a given word is the correct one.

<table>
<thead>
<tr>
<th>Language</th>
<th># of tokens</th>
<th>POS</th>
<th>fPOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swedish</td>
<td>20377</td>
<td>96.27%</td>
<td>94.13%</td>
</tr>
</tbody>
</table>

Table 5.1: Syntaxnet Swedish language accuracy.

The part-of-speech tagging step is crucial in the Natural Language Processing pipeline since the output of this task will provide grammatical meaning to a given sentence. The input of this step will be the output of the previous one.

The Figure 5.9 shows an example of the output produced by the tagger environment for the example sentence.

This output represents a table containing (ordered from left to right) [53]:
1. **ID**: Word index, integer starting at 1 for each new sentence; may be a range for tokens with multiple words.

2. **FORM**: Word form or punctuation symbol.

3. **LEMMA**: Lemma or stem of word form.

4. **UPOSTAG**: Universal part-of-speech tag drawn from our revised version of the Google universal POS tags [55].

5. **XPOSTAG**: Language-specific part-of-speech tag; underscore if not available.

6. **FEATS**: List of morphological features from the universal feature inventory or from a defined language-specific extension; underscore if not available.

7. **HEAD**: Head of the current token, which is either a value of ID or zero (0).

8. **DEPREL**: Universal Stanford dependency relation to the HEAD (root iff HEAD = 0) or a defined language-specific subtype of one.


10. **MISC**: Any other annotation.

### 5.2.3. Syntactic dependency parsing

Syntactic dependency parse is the task that provides the syntactic structure of a sentence concerning relations between words. As an example, a verb will be linked to its dependents: arguments and modifiers. These
relations form a tree-like directed acyclic graph representing the whole syntactic analysis of a sentence. Dependency parsing is also a crucial task in the process of building the knowledge graph. Understanding how actions (e.g. verbs) relate to their arguments and modifiers gives a clue on where the information, within a sentence, may be located.

Universal Dependencies [53] lists 37 universal syntactic relation represented in Table 5.2. This is a revised version of the relations described in the Universal Stanford Dependencies: A cross-linguistic typology (de Marneffe et al. 2014).

<table>
<thead>
<tr>
<th>Nominals</th>
<th>Clauses</th>
<th>Modifier words</th>
<th>Function Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core arguments</td>
<td>nsubj obj iobj</td>
<td>csubj ccomp xcomp</td>
<td></td>
</tr>
<tr>
<td>Non-core dependents</td>
<td>obl vocative expl dislocated</td>
<td>advcl advmod discourse</td>
<td>aux cop mark</td>
</tr>
<tr>
<td>Nominal dependents</td>
<td>nmod appos nummod</td>
<td>acl amod</td>
<td>det clf case</td>
</tr>
</tbody>
</table>

Table 5.2: Main universal syntactic dependency relations [53].

The table is divided into two parts: the upper part follows the primary organising principles of the taxonomy: rows represent relations to the head (the root of the tree), and columns represent basic categories of the dependent.

<table>
<thead>
<tr>
<th>Coordination</th>
<th>MWE</th>
<th>Loose</th>
<th>Special</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>conj cc</td>
<td>fixed flat compound</td>
<td>list parataxis</td>
<td>orphan goeswith reparandum</td>
<td>punct root dep</td>
</tr>
</tbody>
</table>

Table 5.3: Auxiliary universal syntactic dependency relations [53].

For this task, the previously introduced tool: Syntaxnet, will produce the tree representing the dependencies as can be seen in Figure 5.10 (A excerpt of the graph that can be fully seen in Appendix A. Also, Figure 5.11 represents more clearly the relations established. Graph’s row direction has been inverted for clarity.
Figure 5.10: Dependency annotated graph.
5.2 Generating information triples

Figure 5.11: Dependency annotated graph.
5.2.4. Named entity recognition

The task of extracting keywords in a given sentence is called Named Entity Recognition. It is a sub-task for identifying names, places, products, locations, organisations and so on. This task is also crucial since, after it, the pipeline produces a more specific graph that represents only those parts of the text that are of interest for extracting information and, later on, generating the knowledge graph.

Implementing a Named Entity Recognition system can be seen as a Machine Learning classification problem. To help with that, the first tool used to implement this classifier is the Stanford CoreNLP Named Entity Recogniser [74], which implements a Conditional Random Field Classifier [137][138], which is a stochastic model commonly used to label and segment data sequences or extracting information from text data.

One of the challenges of this type of Machine Learning classifiers is the amount of data they need for training. In order to generate enough labeled data, Prodigy\(^2\) [76] (an interactive tool for labelling text data for Machine Learning) is used. The Figure 5.12 shows part of the interface for training data for the Named Entity Recognition model.

![Figure 5.12: Train data labelling using Prodigy.](image)

Figures A.3 (a excerpt of the full graph available in Appendix A) and 5.14 shows the output of this step, where the previous graph has been completed with all the identified named entities.

\(^2\)Prodigy: https://prodi.gy
Figure 5.13: Named Entity Recognition graph.
5.3 Integrating information triples

To evaluate the accuracy of the proposed Named Entity Recognition model, several measures have been defined. Token-level accuracy is one of the available measures, but it has two problems: most tokens in human-generated texts are not part of named entities, which means the probability of a system to classify “not an entity” is very high, usually greater than 90%; and an error in the delimitation of a named entity is not adequately penalised. For example, detecting only the first name of the person when his last name is after that it is scored with half the accuracy.

In academic conferences such as CoNLL[43], a variant of the F-value has been defined as follows:

- **Accuracy** is the number of named entities that exactly match the evaluation set.

- **Recall** is the number of entities in the evaluation set that appear exactly in the same position in the predictions.

- **F-Value** is the harmonic mean of the Accuracy and the Recall. It follows from the above definition that any prediction that misidentifies a token as part of a named entity or stops detecting a token that is a named entity or misclassifies it will not contribute to accuracy nor recovery.

5.2.5 Relation extraction

The final task related to generating information triples is the relation extraction. It consists of finding relations between words in the given sentence. To do so, Stanford CoreNLP provides a Machine Learning algorithm for extracting relations, based on global inference via linear programming formulation [139]. The relations extractor has been trained with custom data representing the most common relationships of the domain being studied in this project.

Figure 5.15 shows one of the triples extracted. It represents a basic unit of knowledge present in the given sentence.

5.3 Integrating information triples

Once the triples have been generated, they need to be integrated to, later on, build the final knowledge graph. This process involves the Stanford Pattern-based Information Extraction and Diagnostics [140] which provides a method for learning entities from an unlabelled text and provides a metric for scoring patterns. This algorithm uses manually defined patterns to extract information that later can be used to integrate the information triples.
5.4 Constructing the knowledge graph

The SPIED algorithm, combined with previously mentioned tools, enables the system to integrate the extracted triples and relate them regarding common topics or elements.

5.4 Constructing the knowledge graph

Finally, when all the triples are integrated and related to each other, the result as can be seen in Figure 5.17 (nodes in yellow represent new information attached to a previous knowledge graph). It is a Knowl-
5.4 Constructing the knowledge graph

![Diagram of SPIED algorithm]  

**Figure 5.16:** SPIED algorithm [140].

...
Figure 5.17: Named Entity Recognition graph.
CHAPTER 6
Evaluation

This chapter presents the evaluation of the implemented solution, including a description of the techniques used for such evaluation and the results obtained.

The technique used for evaluating the system consists of an assessment of the correctness of the knowledge graph generated. To do so, fifty text documents extracted from the Telia’s support forum\(^1\) were prepared to be processed by the pipeline. This preparation consisted in manually extract all the knowledge and information considered representative of those texts and manually build example knowledge graphs.

After the preparation, the same texts were processed using the developed pipeline. The output knowledge graphs were stored together with the manual generated ones to be evaluated regarding entities extracted and relationships generated.

The primary metrics chosen for evaluating the effectiveness of the pipeline are the Precision and Recall. They measure the extent to which the system produces the appropriate output (recall) and only the appropriate output (precision), in other words, completeness and correctness.

\[
\text{Precision} = \frac{\text{correct}}{\text{correct} + \text{incorrect}} \quad (6.1)
\]

In this case, \(\text{correct}\) are the number of correctly identified and extracted relations and \(\text{incorrect}\) the number of errors produced.

\[
\text{Recall} = \frac{\text{correct}}{\text{total}} \quad (6.2)
\]

In case of the Recall, \(\text{total}\) expresses the number of expected relations and entities to be represented in the knowledge graph generated.

To obtain a more fine-grained measure of the performance of the system, the f-measure is used as a weighted harmonic mean of precision and recall.

\(^1\)https://forum.telia.se
\[ F_{\text{measure}} = \frac{(\beta^2 + 1 \times \text{precision} \times \text{recall})}{(\beta^2 + 1) + \text{recall}} \] (6.3)

In this case, beta is a non-negative value that adjusts the relative weighting for recall and precision.

The results obtained in the evaluation of the system obtained a precision and recall values of 0.88 and 0.90 respectively, with an f-measure of 0.88.

These metrics represent the correctness of the Knowledge Graph generated by the automatic pipeline presented in this document. To calculate them, the process followed use two different versions of the same Knowledge Graph: one which is the actual graph generated by the pipeline and another which is generated manually by humans performing the tasks of knowledge extraction and representation.

![Figure 6.1: Pipeline generated graph.](image1)

![Figure 6.2: Manually generated graph.](image2)

Once the two graphs are in place, they are compared between each other regarding the two elements they represent: nodes and links. The so-called Golden Standard in this calculation is the manually generated graph, which is considered, to all extent, correct. For each of the elements (nodes and links) being compared, if there is an inconsistency between both graphs, it is considered an error (incorrect) and will affect the precision and recall as described in the formulas previously on this chapter.

To exemplify this, Figure 6.1 represent a sample graph that let’s suppose was generated by the pipeline. On the other hand, Figure 6.2 represent the manually generated one. As we can see, there is an error on the graph comparison, since the manually generated graph includes a link between two nodes that is not present in the pipeline graph.
CHAPTER 7

Conclusions and Future Work

7.1 Conclusion

The primary goal of this thesis is to implement a system capable of, using human-generated text in Swedish, extract the essential information from it and build and maintain a Knowledge Base with all of the data, not only provided by the input text but by further input that the system will eventually receive.

Current developments in the field of Knowledge Representation had proved to be very valuable not only to the field of Artificial Intelligence, but also other related fields that demand a massive amount of information in a format that can produce valuable information through reasoning using the information stored. Unfortunately, creating and maintaining Knowledge Bases using classical approaches is a time consuming task that prevents its further development and penetration into real use cases.

Automating the task of creating and maintaining those Knowledge Bases is one of the current challenges not only in the field of Knowledge Representation but also automatic reasoning. Being able to have a system capable of generating a Knowledge Base that is self-sufficient concerning adding new information to it, is one of the most compelling work areas that the mentioned field has.

Machine Learning has proven its capability of solving complex problems that humans never believed a computer could solve before. Image recognition or language understanding are just examples of what this technology is capable of. Because of that, Machine Learning is a framework worth testing for trying to narrow the current gap of automation that Knowledge Representation is having. Because of that, this thesis focuses on testing Machine Learning algorithms as a tool for solving the problem stated, as the research question identified states: How can ML algorithms for NLP and Graph Databases be integrated and applied
to build a Knowledge Graph using domain-specific human-generated Swedish text?

As a result of this problem description, a Machine Learning (or more specifically, a Natural Language Processing) pipeline is implemented to test the capabilities of this sort of techniques that Machine Learning provides. This pipeline consumes raw text data and, using different transformation steps, extracts valuable information from that text in the format of triples that represents knowledge. Once those knowledge triples are generated, they are combined in a Knowledge Base that represents information about a domain.

After all, the results of the implemented pipeline show that indeed Machine Learning can be used as a way of automating the task of creating and maintaining Knowledge Bases.

### 7.2 Discussion

The present project concludes reaching the goal of implementing a system capable of building knowledge bases in the format of knowledge graphs using an NLP pipeline that integrates state-of-the-art algorithms and methods for extracting information enclosed in Swedish human-generated text. The research conducted, the literature review and the approach used proves the feasibility of this kind of systems for automate the task of constructing and maintaining structures for knowledge representation.

Now, to close the learning cycle started with the development of this work, and with the aim of highlighting the lessons learned, it is necessary to finish this report with a critical reflection on the work done.

Natural language is the mean we use on a daily basis to establish our communication with other people. The richness of its semantic components gives natural languages their great expressive power and their value as a tool for subtle reasoning. However, this valuable characteristics of natural language are challenges for the field of Natural Language Processing, that is, natural language has properties that reduce the effectiveness of textual information retrieval systems:

- They are developed by progressive enrichment before any attempt to form a theory.
- The linguistic variation creates the possibility of using different words or expressions to communicate the same idea.
- The linguistic ambiguity makes possible to have more than one interpretation for the same piece of text.
The last two properties affect the information retrieval process differently. Linguistic variation causes information silence, that is, the omission of relevant information to cover the need for knowledge since the same terms that appear in the document have not been used. On the other hand, ambiguity implies documentary noise, that is, the inclusion of information that is not significant, since information that uses the term but with a different meaning are also recovered. These two characteristics make the automated processing of the language a major challenge yet to be solved.

In general, the field of NLP, and more specifically Knowledge Extraction, is still evolving and it is essential to take into account that it did not reach the plateau yet. This means that improvements are to be expected shortly; thus better approaches for solving this problem will undoubtedly appear and may invalidate partially the work done so far. As an example, most of the current projects related to KE are limited to the extraction of binary relations, in the boundary of a sentence. This does not allow to derive more complex structures [66] [141] which can represent information with higher fidelity, which would ease the process being described in this thesis.

Moreover, it is important to mention that the only way to get a real reasoning system based on information and knowledge, is to enable it to store ideas in its "memory" so that it can manipulate them and use them when needed. However, what is sought is the ability of machines to represent ideas.

Currently, the main Machine Learning techniques are teaching computers to represent languages and simulate reasoning (not to reason). A "model of ideas" that is independent of language is needed. The interface with that model would be a kind of translator between its language and the machine ideas model language. However, very little is being accomplished in this area. One option is to build a new model of thinking for computers manually. This would require a massive collaboration among computer scientists, linguists, neuroscientists, philosophers, mathematicians and others.

Still, it seems that it is a problem that needs to be tackled eventually. Once a model of ideas is available, instead of a language, a real reasoning system may be created. Not only will it learn associations between words like a chatbot, but it will be able to draw conclusions after combining ideas. It will take much work to achieve this level of generalised reasoning, but what it is the truth is that continuous work on Knowledge Bases and Knowledge Representation is needed to get close to that idea.
7.3 Future work

The perspective of the developed solution, already implemented, is quite encouraging, and can be extended to other fields or use cases, as well as other entities that would like to use this technology. However, in order to improve and expand the work done, several lines of work are proposed for the future:

**Enriching the Knowledge Graph using external sources**
Using external data sources such as DBpedia[142], the knowledge graph can be filled with even more facts and interesting information.

**Accuracy improvement**
Due to time limitations, accuracy was not one of the main focus points to implement this solution. That is why one of the most important future work tasks is to try different algorithms and configurations to improve accuracy.

**Scaling**
Another interesting area for future work is to measure how well the knowledge graph can scale when it contains a vast amount of facts and relationships.


[15] Xuedong Huang, Alejandro Acero, Alex Acero, and Hsiao-Wuen 
Hon. Spoken language processing: a guide to theory, algorithm, and 

[16] Alvaro L Fraga, Marcela Vegetti, and Horacio Leone. Semi-
Automated Ontology Generation Process from Industrial Product 

[17] David W. Embley, Douglas M. Campbell, Randy D. Smith, and 
Stephen W. Liddle. Ontology-based extraction and structuring of 


[20] Open Working Group, Sustainable Development Goals, 
Open Working Group, Sustainable Development Goals, Develop-

[21] Anne Håkansson. Portal of Research Methods and Methodolo-


[24] G. Leitner. Swedish as a pluricentric language. Pluricentric lan-


BIBLIOGRAPHY


[93] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. Automated Phrase Mining from Massive Text Corpora, Feb 2018.


APPENDIX A

Graphs
Figure A.1: Part-of-speech and syntactic dependencies graph.
Figure A.2: Dependency annotated graph.