UNIVERSIDADE DO ALGARVE
Faculdade de Ciências Humanas e Sociais

THE UNIVERSITY OF WOLVERHAMPTON

School of Law, Social Sciences and Communications

Relation Extraction for People Search on the Web

Tilia Ellendorff

Mestrado Internacional em Processamento de Linguagem Natural e Indústrias da Língua

International Masters in Natural Language Processing and Human Language Technology

UNIVERSIDADE DO ALGARVE
Faculdade de Ciências Humanas e Sociais

THE UNIVERSITY OF WOLVERHAMPTON
School of Law, Social Sciences and Communications

Relation Extraction for People Search on the Web

Tilia Ellendorff

Orientador/Supervisor: Constantin Orasan
(University of Wolverhampton, United Kingdom)
Orientador/Supervisor: Nuno Mamede
(Universidade Téchnica de Lisboa, Portugal)

Dissertation submitted as part of the study program for the award of the Master degree in Natural Language Processing and Human Language Technology, and supported by a grant from the European Commission, Education & Culture, under the Erasmus Mundus Master Courses Program (ref. EMMC 2008-0083).

Date of Submission : 14 May 2012

FARO, 2012

UNIVERSITY OF WOLVERHAMPTON SCHOOL OF LAW, SOCIAL SCIENCES AND COMMUNICATIONS

MA NATURAL LANGUAGE PROCESSING & HUMAN LANGUAGE TECHNOLOGY

Name:____

Date:__________________________

Title:____

Module Code:______________________

Presented in partial fulfilment of the assessment requirements for the above award

Supervisors:____________________ **Declaration:**

*This work or any part thereof has not previously been presented in any form to the University or to any other institutional body whether for assessment or for other purposes. Save for any express acknowledgements, references and/or bibliographies cited in the work, I confirm that the intellectual content of the work is the result of my own efforts and of no other person.

It is acknowledged that the author of any project work shall own the copyright. However, by submitting such copyright work for assessment, the author grants to the University a perpetual royalty-free licence to do all or any of those things referred to in section 16(i) of the Copyright Designs and Patents Act 1988 (viz: to copy work; to issue copies to the public; to perform or show or play the work in public; to broadcast the work or to make adaptation of the work.

*This project did not involve contact with human subjects, and hence did not require approval from the LSSC Ethics Committee.

Signed:__________________________________ Date:_____________

Abstract

The topic of this dissertation is a relation extraction system for people search on the web. Nowadays, a big amount of search queries on the internet are about people. In general, people can be described by their attributes, such as profession, place of birth, or places where they lived. The present work focuses on sentences which express the relation between a person and a respective attribute. It adapts a mechanism of distant supervision, developed by Mintz et al. (Mintz et al., 2009), which has the underlying intuition that a sentence containing two entities which are known to stand in a certain relation to each other, is likely to express this relation in some way. Based on this intuition, a sentence extraction system is built which is used to accumulate training data for possible systems of relation extraction for people search on the web. The sentence extraction systems is based on scripts in the programming language Python and it uses Wikipedia and Freebase as resources. Freebase provides entities which are known to stand in a specific relation to each other and Wikipedia is used for discovering sentences which fulfil the conditions stated by the underlying intuition of distant supervision, and extracting them. The attributes that are examined are profession, nationality, place of inhabitance of a person and places where a person lived. Manual analysis of small samples of the obtained data will on the one hand show how effective the method is for the chosen attributes and on the other hand be used to discuss possible ways of making the data less noisy. Furthermore, an example system is built within Python and experiments are carried out in order to show what impact slight changes in the sentence extraction system can have on machine learning systems for relation classification. One point of interest of the presented dissertation, is to investigate into a relatively easy and intuitive approach for relation extraction in order to explore its weaknesses and strengths. The second, and more practical aim, is to present and test ideas which can possibly make a distant supervised system more effective.

Resumo

Na Web, existem grandes quantidades de informação não-estruturada relativa a indivíduos. Ler toda essa informação, até que se pudesse saber mais acerca de uma pessoa específica, tomaria muito tempo. Enquanto, no passado, seria necessário pesquisar um grande número de websites, individualmente, para que se encontrasse a informação pretendida, este consumo de tempo pode agora ser evitado pelo uso de uma aplicação informática que desempenha a mesma tarefa mais rapidamente e de forma automática. A informação pode, depois, ser armazenada em bases de dados para sua posterior utilização para os mais diversos fins.

Uma altíssima percentagem de pesquisas efetuadas na Internet, é, hoje em dia, relativa à busca de informação acerca de pessoas. Aquilo que torna uma pessoa diferente de outra são os atributos pelos quais ela pode ser descrita, tais como: a sua profissão, a sua origem, a sua língua, pessoas com quem se relaciona de alguma forma, entre muitos outros. Estes atríbutos, para além de serem responsáveis por distinguir uns indivíduos dos outros, são também, frequentemente, o alvo de quem pesquisa por informações relativamente a pessoas.

Quando se analisam as frases que expressam os atributos de uma pessoa, como por exemplo a profissão, é visível a existência de uma relação binária entre duas entidades: por um lado a pessoa e por outro lado a profissão. O objetivo da extração da relação (relation extraction) é extrair as partes que se relacionam umas com as outras.

O tópico desta dissertação é a supervisão remota de sistemas de extração para pesquisa de pessoas na Web. O foco particular deste trabalho é o mecanismo de supervisão remota, desenvolvido por Mintz et al. (Mintz et al.,2009). Nesse trabalho, é abordada a construção de um sistema de supervisão semanal com a intuição subjacente de que uma frase contendo duas entidades que se saiba de antemão que se relacionam entre si, expressará, provavelmente, essa mesma relação. Assim, a base de dados, fornece entidades participantes numa relação específica, que podem, por sua vez, ser utilizadas para um tipo de supervisão remota. A informação ruído (noisy data), que consiste em frases que apresentam condições postuladas na intuição subjacente, são usadas como

dados treino para osistema de aprendizagem computacional. Mintz et al. referem que a utilização extensa de dados de treino pode ajudar na abordagem ao ruído.

Baseado no método de Mintz et al., um sistema de extração de frases foi construído, alicerçando-se no conceito de intuição subjacente para acumulação de dados de treino, na perspetiva de construir sistemas de extração de relação para pesquisa de pessoas na Web. O método do sistema é extrair as frases que contém o nome da pessoa, bem como, o respetivo atributo. O sistema de extração de frases é baseado em scripts em línguagem de programação do Python e usa a Wikipedia e o Freebase como recursos. A Freebase fornece entidades que, se sabe à partida, relacionarem-se especificamente entre si. A Wikipedia foi utilizada para obter frases que cumpram as condições postuladas pela intuição subjacente da supervisão remota, e extrai-las. O sistema de extração de frases foi construido para os seguinte quatro atributos: profissão, nacionalidade, local de habitação de um indivíduo e locais onde essa pessoa viveu.

Pequenas amostras dos dados obtidos pelo sistema de extração de frases foram manualmente analisadas, no sentido de apurar possíveis erros e para discussão de características dentro dos dados. Isto foi feito, por um lado, com o objetivo de mostrar a efetividade de cada um dos atributos escolhidos e, por outro, para apurar possíveis estratégias de redução do ruído nos dados.

Algumas das formas propostas para redução do ruído foram testadas num sistema de aprendizagem computacional exemplo, com vista à classificação da relação no que concerne ao atributo "nacionalidade". Das frases, originidas pelo sistema de extração, foram extraídas características, nas quais, um classificador Naïve Bayes, foi treinado. As características extraídas são de ordem: léxica, tendo como base características bag-of-words, bigram e parte das etiquetas de discurso. A extração de caracteristicas e o treino do classificador foram feitos com auxílio do NLTK (Natural Language Processing Toolkit for Python – Ferramentas de Processamento de Linguagem Natural para Python).

Utilizando este exemplar de sistema, foram feitas experiências no sentido de mostrar o impacto de mudanças ligeiras no sistema de extração de frases, pretendendo-se assim, fazer com que os dados contenham menos ruído. Um dos pontos de interesse desta dissertação é procurar uma aproximação fácil e intuitiva à extração de relações, de modo a explorar as suas potencialidades e as suas fragilidades. Um segundo objetivo, mais prático, é o de apresentar e testar ideais que possibilitem tornar um sistema de supervisão remota mais efetivo.

Acknowledgements

First of all, in relation to the presented work, I want to express my gratitude and appreciation to my supervisors, Constantin Orasan and Nuno Mamede, working together with who was a great pleasure and without their support and guidance this work would not have been possible.

Furthermore, I want to express my gratitude especially to Pedro Paulo Balage for your kind helpfulness and support, whenever I needed it. I appreciate very much the time that you spend for giving me advice concerning my project and providing explanations to all my questions. Additionally, I want to thank Iustin Dornesco, for giving initial advice and guidance and for helping me in finding ideas based on which this project came into existence and for providing the Wikipedia data dump, and Wilker Aziz for his friendly help providing additional computational resources and support with running data on the server.

My appreciation goes to the master's consortium for choosing me for the program and the scholarship, without which it would not have been possible to make the valuable experience of living abroad and studying within an international, multicultural environment at University of Wolverhampton and Universidade do Algarve.

Having completed the Master Mundus program “International Masters in Natural Language Processing and Human Language Technology”, looking back I can conclude that these were two of the most important and best years of my life. I have not only succeeded in extensively expanding my knowledge in an area that I genuinely enjoy, but I have also gotten to know some of the best people that I ever met. For this reason, finally, I want to thank all my best friends which I made during these two years of studying together: Maja, Eleni, Anya, José Guilherme, Ruth, Ekaterina and Pamela. Additionally, I would like to thank my English and Portuguese friends which I have met in the two countries where I spent the last two years.

Last but not least, I want to express my deepest gratitude to my parents for being always there for me, supporting me in my plans and my ideas and investing a lot of time and money in my education.

Contents

1 Introduction 1

1.1. Extracting Attributes of People from unstructured Text 1

1.2. Relation Extraction and Weakly Supervised System 3

1.3. Overview over the Present Project

2 Relevant Literature on Information Extraction and Relation Extraction 6

2.1. Information Extraction in General 6

2.2. Relation Extraction 8

2.2.1. Knowledge Engineering Systems for Relation Extraction 9

2.2.2. Supervised Systems for Relation Extraction 10

2.2.3. Weakly Supervised Systems for Relation Extraction 11

2.2.3.1. Bootstrapping Systems 11

2.2.3.2. Distant Supervision 13

2.2.4. Unsupervised Systems for Relation Extraction 15

2.3. The Linguistic Background of Relation Extraction 16

2.4. Dataset: Wikipedia and Freebase 18

2.4.1. Wikipedia 18

2.4.2. Freebase 20

2.5. Conclusions of the Literature Review for the Present Work 20

3 The Sentence Extraction System 22

3.1. Data Set and Chosen Attributes 22

3.2. Methodology of the Sentence Extraction System 24

3.3. Analysis of the obtained Data for the chosen Attributes 27

3.3.1. Places Lived 28

3.3.2. Place of Birth 32

3.3.3. Profession 35

3.3.4. Nationality 38

3.4. Conclusions of the Manual Analysis 43

4 Methodology of the Machine Learning System 45

4.1. NLTK: the Natural Language Toolkit for Python 46

4.2. Naïve Bayes Classifier 46

4.3. Data for Training and Testing 48

4.4. Preprocessing of Data and Selection of Features 49

4.5. Results and Evaluation 51

4.6. Possible Improvements of the System 53

5 Final Conclusion and Ideas for Future Research 56

6 Appendix 59

6.1. Example Script: Sentence Extraction System 59

6.2. Script: Machine Learning System 63

7 Bibliography 71

7.1. References 71

7.2. Resources 76

1 Introduction

On the web there are large amounts of unstructured information about people. It would take a huge amount of time to read through all of them in order to find out more about a specific person. The primary aim of information extraction is to find a way to get to the needed information fast and easily. This can be useful where ever there are large amounts of unstructured text in electronic format. Whereas in the past, a lot of time was necessary to search through many webpages in order to finally find the wanted information, this expenditure of work and time can be avoided by the use of an application which can do the same thing faster and automatically. Subsequently, the information can be stored in databases and be used for many different purposes.

Google insights[1], a database which stores statistics of Google search queries in the past, shows that a high percentage of search enquiries on the Internet nowadays is looking for information about people. What makes one person different from other people are the attributes by means of which a person can be described, such as profession, origin, people with who a person is related in some way, languages, among many more. Apart from making one person unique and different from others, attributes are the wanted information in most cases of people search on the Internet. By finding ways for extracting the attributes of people automatically, the process of finding the wanted information can be accelerated and facilitated for a potential user.

1.1 Extracting Attributes of People from unstructured Text

When trying to extract the attributes of people, sentences which contain the wanted information could have the following format:

1. *"Foerster was hired by the San Francisco 49ers as the co-offensive line coach."*[2]
   (Foerster, coach) – relation-type: profession

2. *"John Luther Adams is a composer whose music is inspired by nature."*[3]
   (John Luther Adams, composer) – relation-type: profession

1 http://www.google.com/insights/search/#cat=0-14&date=1%2F2010%2029m&cmpt=q
2 http://en.wikipedia.org/wiki/Chris_Foerster
3 http://en.wikipedia.org/wiki/John_Luther_Adams

In these sentences, which are taken from Wikipedia articles, the attribute “profession” is expressed for two different people. There are two elements in each sentence, which stand in a relation: For the first sentence the name of the person “Foerster” and the profession “coach”, for the second sentence the name “John Luther Adams” and the profession “composer”.

If several sentences containing a certain relation between a person and a connected attribute are available, they can be analysed linguistically with regard to patterns and structures which they have in common. The aim of such an analysis is to find common structures which are characteristic environments for the expression of a certain relation. Within this scenario, the more sentences are available for a certain relation type, the higher is the probability that all patterns and structures of all kind of sentences containing a certain attribute (in this case “profession”) are taken into account. After several of these specific structures, have been collected, they can be used with unstructured text in order to find other sentences containing the same relation type. In other words, they can be applied for classifying sentences according to whether they contain a specific relation or not. If a new sentence has been classified correctly, the respective lexical items, which partake in the identified relation, can be extracted.

In this regard, the task of relation extraction, in most cases, consists of two sub parts: relation detection, which is normally done by classifying sentences according to whether they contain a specific relation type or not and, subsequently, the actual extraction of the relation, which consists in extracting the elements involved in a relation once a sentence has been classified correctly. In the whole process, the first part is crucial, since without identifying a sentence as containing a relation of a specific relation type, the extraction of the individual elements would not be possible. The focus of the present project therefore is on relation detection as the first part of the process.

Even though the present project focuses on people's attributes, relation extraction is used for all kind of different types of relations, such as relations between places and objects, relations between books and their genres or even semantic relations between words such as synonymy, hypernymy or metanymy.

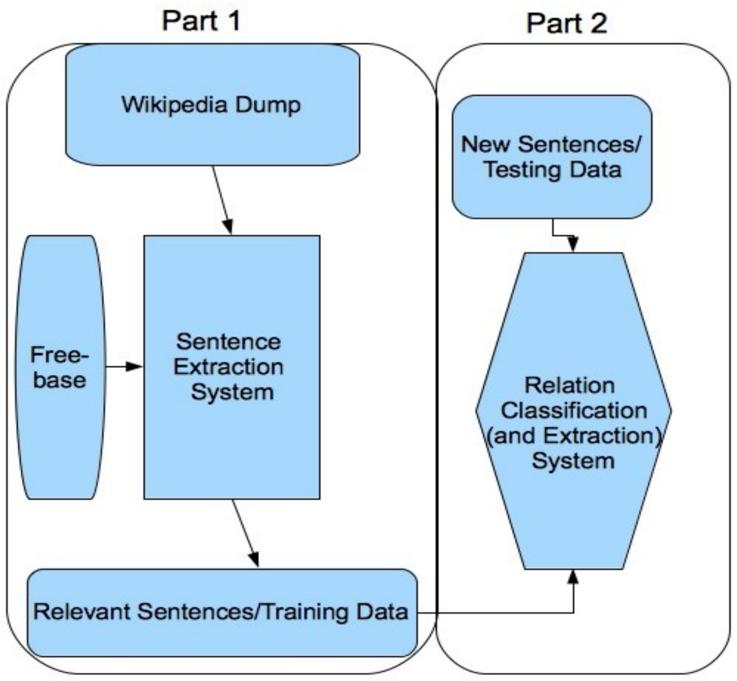
1.2 Relation Extraction and Weakly Supervised Systems

In the last 50 years, a lot of research has been going on in the field of relation extraction, which has been further strengthened by the growth of the Internet as a vast collection of unstructured text during the last 15 years. In the beginning, systems for relation extraction were mainly rule-based, using hand-crafted rules for discovering relations and extracting them. When machine learning became more popular, supervised systems for relation extraction became fashionable. However, both types of these systems require a certain amount of manual effort: Rule-based systems in crafting the rules, and supervised systems in manually annotating tagged corpora which serve as training data for the machine learning algorithms. One aim of current research in relation extraction is to exchange the involvement of human manual effort for efficient automatic methods as far as possible. Automatic methods can make a system simpler, less expensive and less time intensive in so far as human manual work generally needs time and money. Until now, however, systems involving human manual work, namely supervised systems, still show a significantly better performance than systems which try to manage with less human involvement. For this reason, research in the area of weakly supervised or unsupervised systems for relation extraction is necessary. Weakly or semi-supervised systems try to find new ways to provide supervision without requiring or only requiring little human manual work but still achieving a high performance.

For the project of relation extraction for people search on the web the focus will be put on the subarea of weakly supervised systems, and on the problem of extracting people's attributes. The project takes as an initial point a method of weak supervision developed by Mintz et al. (Mintz et al., 2009), which will be described in detail in section 2.2.3.2 of the present work. Based on the underlying intuition of this approach, that each sentence containing a name of a person and an attribute which are previously known to stand in a certain relation type to each other, is likely to express this relationship, training data for four different attributes will be extracted and analysed. This will be done in the first part of the project, described in chapter 3. Finally, in chapter 4, experiments will be carried out in order to test suggestions of improvement of the underlying intuition which are expected to make the training data less noisy.

1.3 Overview over the Present Project

The system of weak supervision, as described in the present project, consists of two major parts: the extraction of sentences as training data as a first part, and the use of this set of sentences for training a classifier which can be applied for classifying unseen sentences regarding if they contain a specific type of relation or not, as the second part. As described above, classifying sentences by using such a classifier is necessary in order to extract the elements which are in relation to each other. An overview of the whole architecture of the current project is given in Flowchart 1.


Part 1
Part 2
Wikipedia Dump
Free-base
Sentence Extraction System
Relevant Sentences/Training Data
New Sentences/ Testing Data
Relation Classification (and Extraction) System

Flowchart 1: Overview over the whole Project Architecture

Within the first part, which is described in chapter 3, a Wikipedia data dump provides the data from where sentences, which are like to contain a specific relation, are extracted. Following the underlying intuition, it is necessary to have seeds available in the form of two entities, for which it is already known that they stand in a certain relation to each other. These entities are provided by Freebase, a large repository of structured data which is freely available on the Internet.[4] With the help of the seed entities, relevant sentences, which fulfil the conditions stated by the underlying intuition, are extracted from the data dump for four chosen attributes. The extracted sentences are then manually analysed. Manual analysis is used in order to evaluate the

4 http://wiki.freebase.com/wiki/Main_Page

method for these specific attributes and to perform error analysis of the sentence extraction system. Furthermore, manual analysis is necessary for discovering features which can be used for the machine learning system within the second part of this project.

The second part of the project is described in chapter 4. The extracted sentences from part one are used as, potentially noisy, training data for training an example machine learning system for relation classification. The purpose of this system is to show the impact of methods for making the training data less noisy, making slight changes on the sentence extraction system. After the system has been trained, it can be used with testing data in the form of new sentences to discover if the system is able to classify them correctly.

Summing up, the objective of the present project will be, focusing on a small selection of attributes, to show whether and in how far the mechanism of distant supervision, developed by Mintz et al., is adequate for extracting relations between people and their attributes and using manual analysis with the aim of making suggestions for improvements to the original mechanism of distant supervision.

2 Relevant Literature on Information Extraction and Relation Extraction

This literature review will present the attempt to circle the topic of relation extraction for people search on the web from different angles in order to situate it in the broader field, first of information extraction in general, and then, more narrowly, relation extraction. To provide a thorough background to the topic, the main approaches of research within the field will be considered. Moreover, literature about the linguistic aspects of relation extraction will be taken into account as well as literature on the data set which, in the course of the present work, will be used for building a system for relation classification.

2.1 Information Extraction in General

In general, the process of information extraction in literature is roughly defined as turning "unstructured information embedded in text into structured data" (Jurafsky, 2007). This means that specific information, which can be found in the unstructured text, is extracted and subsequently presented in a more structured and therefore more understandable layout. Furthermore, structured knowledge can be stored in databases in used for different purposes. Comprehensive overviews over the field of information extraction were written by Grishman (2003) and (2010) and by Jurafsky and Martin (2007) as chapters within books about Natural Language Processing and more elaborately by Moens (2006).

First considered in the research done by Harris (Harris, 1958), more than fifty years ago, (comp. Grishman, 2010), the field of information extraction got more popular in the 1980s and finally research was promoted through the seven Message Understanding Conferences (MUC), of which the first took place 1987 and the last in 1996. Research in the field, however, was biased by the tasks which were released in line with these conferences (comp. Grishman, 2010). A short history and overview of the seven MUC conferences can be found in Grishman and Sundheim (1996).

Following MUC, the ACE (Automatic Content Extraction) workshops were held as well as other workshops and conferences focusing on special subareas such as the WePS

(Web People Search) workshop. Furthermore, information extraction tasks were released in line with conferences and workshops with a broader spectrum such as SemEval (Semantic Evaluation Workshop). Different information extraction tasks are named entity recognition (NER), coreference resolution, relation extraction (also called relation detection and classification), event extraction (also called event detection and classification) and temporal extraction (also called temporal expression detection) (Comp. Jurafsky, 2007). Of all of these, NER is the task for which most research has been done so far. As the project of people search on the web focuses on relation extraction, the literature of this subfield will be explored more thoroughly later in this literature review.

Information extraction systems can be roughly categorized into four different kinds of systems: knowledge engineering systems (also rule-based systems), supervised systems of machine learning, unsupervised systems of machine learning and weakly supervised systems of machine learning (Comp. e.g. Grishman, 2010).

In many cases, knowledge engineering systems are being considered as not very up-to-date as they make large amounts of hand-crafted rules necessary. Although they show an overall good performance, the amount of time and work that has to be invested is huge, which occasionally renders them less practical. They require linguistic experts which have a very thorough understanding of the linguistic context of the information which is to be extracted by the knowledge engineering system. The advantage of rule-based systems, however, is that they are very clear and flexible, as rules can be added and adapted as needed (comp. Sarawagi, 2007). As the rules have been developed by linguists, they are based on specific knowledge, which makes them very precise. Furthermore, rule-based systems show an advantage in terms of speed in many cases (comp. Sarawagi, 2007). Although the first systems for information extraction that existed were pure knowledge engineering systems based on hand-coded rules (comp. Grishman, 2010), today, in many cases, knowledge-based approaches are used together with supervised machine learning algorithms.

Systems based on supervised machine learning generally need large amounts or labelled data for training and testing which again needs time and effort. Furthermore, supervised systems are likely to suffer from overfitting which is another disadvantage.

Overfitting happens when the supervised system during training, gets too well fitted to a training set, to a degree that it reaches very high results with this specific set but shows a far worse performance with all other kinds of testing sets.

Unsupervised systems are based on the idea of not using any kind of labeled data, but relying solely on similarities of certain linguistic structures which makes it possible to build clusters of similar structures. However, these systems are usually less accurate and therefore have not been very widely used in the past.

Finally, weakly supervised systems have become very popular in the last years and represent the most promising and most modern type of systems for information extraction. As the decision has been made to focus on weakly supervised systems for the research on relation extraction for people search on the web, literature about these types of systems will be considered in more detail under the subtopic of relation extraction.

2.2 Relation Extraction

Relation extraction (also: relation detection and classification, comp. Jurafsky and Martin, 2007) is the subfield of information extraction which deals with discovering and extracting relations between entities in unstructured text. In most cases, relation extraction can be regarded as a classification task, as a first step which is followed by an extraction task.

When trying to extract, for example, different attributes of people from a text, relation extraction is necessary: Looking more closely at the nature of peoples' attributes in general it becomes clear that they all describe a kind of relationship between a person and another entity, as shown by means of examples in the introduction. This other entity can, for example, be a location (e.g. country of inhabitance), organization (e.g. employer) or other person (e.g. marriage). In most cases, these type of relations can be described as binary relations. Apart from looking for attributes of people, relation extraction is used within several other more general tasks of natural language processing, such as, for example, question answering, biomedical information extraction and ontology population.

A survey of relation extraction, although already a few years old, can be found by Bach and Badaskar (2007). Grishman (2010) in the context of relation extraction terminology makes a difference between the notions of *relation* and *relation mention. Relation,* according to this terminology, expresses the general concept of relationship between two entities, as for example "nationality" or "profession" and *relation mention* is used for the specific occurrence of a relationship in a text. In the sentence "Claridge Manuela Kasper is a journalist from Germany", a *relation mention* between Claridge Manuela and Germany can be identified. This difference has been widely accepted in literature.

As relation extraction is a subfield of information extraction, relation extraction systems generally can be divided into the four different classes of information extraction systems which were stated above. Knowledge engineering systems will only briefly be considered, along with supervised systems and unsupervised systems and more thorough focus will be put on weakly supervised systems, as these are the main interest for the present project of relation extraction for people search on the web.

Furthermore, systems of relation extraction can be divided into traditional systems for relation extraction and open systems of relation extraction as defined by Banko and Etzioni (2008). Traditional systems are relation-specific systems which make a previous definition of relations necessary and which are designed to look only for these relations and nothing else. Open systems for relation extraction, on the contrary, are relation independent and designed to extract all kinds of relations without predefining them.

2.2.1 Knowledge Engineering Systems for Relation Extraction

In line with the majority of first information extraction systems, the first systems which could perform relation extraction were based on a knowledge engineering approach using hand-coded extraction rules. Rules are normally based on lexical and syntactical patterns and often coded into the form of regular expressions which can identify the sentential environment in which normally a relation is present. This is done with the aim of extracting the pair of entities which are connected through a predefined relation type.

For this purpose, a collection of rules has to be developed for each relation type that is supposed to be extracted.

Even though rule-based systems have already been applied for more than 50 years, they are still popular nowadays as they have a very high advantage concerning speed and precision. However, rule-based systems can easily be adapted to optimizations (Sarawagi, 2007) as new rules can be added later without problems. In most cases, rule-based systems consist of two parts: a collection of rules and a set of policies to control how and in which order the rules are fired (Sarawagi, 2007).

2.2.2 Supervised Systems for Relation Extraction

Supervised systems are typically traditional systems for relation extraction, which are built to only extract predefined types of relations. They can be described as classification systems which are designed to perform a classification, according to whether a relation between two entities is present or not. For this purpose, a classifier is trained which needs an annotated corpus, in most cases with positive and negative examples of different given relations. Depending on whether this classifier is trained using a set of features, or takes as input parse trees or other rich structural representation, supervised systems can be divided into feature based methods and kernel methods (comp. Bach and Badaskar, 2007).

Furthermore, different approaches consider different ways of describing the sequence of words between two entity mentions, which can be done, for example, as a sequence of chunks or the path in a parse-tree (comp. Grishman, 2010). It has been shown, however, that successful systems combine the evidence of several representations, as each type or representation has its own benefits, which can make up for the weaknesses of other types of representations (comp. Grishman, 2010). Some influential research on on feature based methods was done by GuoDong et al. (2002), Kambhatla (2004) and Zhao and Grishman (2005). In Kernel methods, the systems of Culotta and Sorensen (2004) and Bunescu and Mooney (2005) as well as Zelenko et al. (2003) can be defined as the major contributions of the field (comp. Bach and Badaskar, 2007). Within the supervised systems for relation extraction, kernel systems generally show a better performance than feature based systems. Although supervised systems in general still perform better than weakly supervised and unsupervised systems, they suffer from other serious disadvantages mentioned above, such as need of labeled data. Moreover they require the use of tools for textual analysis, such as POS-taggers, parsers and dependency parsers.

2.2.3 Weakly Supervised Systems for Relation Extraction

The first weakly supervised system was developed in 1992 by Hearst (1992) and since then weakly supervised (also: semi-supervised or lightly supervised) systems for relation extraction have become popular, mostly within the last 20 years. Currently, weakly supervised systems are considered as the most modern kind of systems for relation extraction and a hot topic within the field of information extraction. They are based on learning algorithms, which do not make large amounts of labeled data necessary as supervised systems do, but normally just need a very small set of labeled data. One more characteristic of weakly supervised systems is that they are normally very fast. Although, in general, their performance is not yet as good as that of supervised systems for relation extraction, research in the last few years has been very concerned with the improvement of weakly supervised systems of relation extraction. This is due to the fact that they are generally considered very promising, especially since they make human manual effort, which is necessary for tagging corpora, expendable. As they require a given set of relations, they can be categorized as relation-specific systems according to Banko and Etzioni (2008).

2.2.3.1 Bootstrapping Systems

The first weakly supervised systems for relation extraction used the simple idea of using regular expressions to extract patterns from the text that are likely to contain a specific relation (comp. Jurafsky and Martin, 2007). This very basic approach was soon replaced by the more interesting and flexible idea of bootstrapping, which became the mainly considered approach of weakly supervised system for relation extraction within the research of the last years. After bootstrapping became popular in natural language processing (e.g. Yarowsky, 1995), the first important bootstrapping system for relation extraction was presented by Brin (1998), who used this method in order to extract relations between books and their authors. This first approach was followed and extended by Agichtein and Gravano (2000) with their influential Snowball system. Newer weakly supervised systems using the main ideas of bootstrapping were developed by Bunescu and Mooney (2007) among others.

Bootstrapping systems enable either to start with single pairs of relations or with patterns in which a certain relation occurs (seeds) in order to extract information about patterns in which a certain relation occurs from unlabelled data. These pairs could be, for example, a certain set of names of people and the places where these people live. On the basis of these pairs, information about possible linguistic patterns (features) is gathered. The gathered information again makes it possible to extract new pairs of relations. A kind of circular information extraction system is formed, which offers the possibility to extract many pairs of people and their related attributes, as well as information about the environments and patterns in which they occur from an unlimited amount of data.

The problem of bootstrapping systems is the semantic drift, which can come along with the very weak supervision of the system combined with the recurrent running of the cycle of relation pairs and patterns: if entities or patterns are not very distinctive for a certain relation, they are very likely to extract features, after some time, which are not necessarily related to the same semantic group to which the seed relation or pattern belonged (comp. Jurafsky and Martin, 2007). For example, if a person lives in Nancy, which is a city in France, but which can also be a given name, the system which is looking for patterns that describe a relation between a person and a location can accidentally extract linguistic patterns which describe a relation between two people. This can lead to the problem that during the following circles of the system more and more features which belong to the wrong kind of relation will be extracted. Semantic drift can be very disturbing, as this means that relation pairs and patterns are extracted that are completely different from the relation and patters which the system was supposed to look for. In order to try and keep semantic drift under control, different approaches of bootstrapping differ, regarding what methods they apply to filter and rank these patterns, and, therefore, decide if they are either distinct and useful or ambiguous and too unspecific for a defined relation (comp. Grishman, 2010).

2.2.3.2 Distant Supervision

In the more recent past, the main idea of bootstrapping has been developed further with newer systems trying to cope with the flaws of bootstrapping systems. One system, which is very interesting for the research of this present project, as mentioned before, was developed by Mintz et al. (2011). Mintz et al. (2011) have presented a way of distant supervision, which is developed with a method in mind that was previously used by Snow et al. (2005). The aim of this method is, among other things, to deal with the problem of semantic drift. One main difference between this approach and the traditional approach of bootstrapping is based on using a limited corpus, as for example a predefined collection of texts, for training the system. This makes the semantic drift at least less prominent than it is with the use of unlimited data. However, it shares some of the advantages of traditional bootstrapping, for example, that it does not need labeled data for training and that it is able to deal with very large corpora.

The distant supervision system of Mintz et al. starts with a given set of 120 relations, which is taken from a database as, for example, Freebase or Dbpedia (in Mintz et al. Freebase and other smaller databases). As was mentioned before, the main underlying intuition of this approach of distant supervision, which is adapted for the present project, is that every sentence containing both named entities of a relation, this relation is expressed in some way.

It can be said that a kind of "supervision" takes places by a database instead of a labeled corpus, as, based on the entries within the database, the training data is chosen. Mintz et al. point out that this brings along more advantages, namely that the system is not as prone to overfitting or domain-dependent as many supervised systems. Before this background, the set of relations are subsequently used to gather information about the context of the occurrence of the two named entities together. The encountered features can then be used again, similar to bootstrapping, to extract new pairs of entities which are connected by the same kind of relation. The preliminary goal in training the system is to build a probabilistic classifier, which contains information about contextual features of certain relations. The features used by Mintz et al. (2010) are syntactical and lexical features in the form of conjunctive features. Conjunctive features, are high precision features, for which, in order to find two matching features, it is necessary that all of their conjuncts match. For this reason, Mintz et al. provide very big amounts of data, more specifically 800,000 Wikipedia articles for training and 400,000 for testing, which are required by these very specific features for discovering matches.

It is important, however, to keep in mind that the features which are extracted by such a system are potentially noisy. But Mintz et al. point out that, as very large amounts of features are extracted and accumulated in the classifier, bad features can be dealt with more easily, as they become more obvious within the classifier. In short, this means that collected evidence from multiple sentences containing a relation pair finally decides if a feature is useful and distinguishing for a given relation, and should be kept, or is misleading, and should be neglected.

The classifier can be used in the following steps with testing data in order to decide if a relation is present in a sentence containing the two entities. Subsequently, a relation that has been detected by the system can be categorized as a certain type of relation within the given set of relations. The output of the system, therefore, is the name of the relation type combined with a confidence score describing the probability that the entity pair belongs to the detected relation type.

Mintz et al. used held-out evaluation as well as human evaluation for measuring the performance of their system. These evaluation techniques measured an average precision of 67.6% while extracting 10.000 instances of 102 relations. With three different runs using either lexical or syntactic features or both of them, Mintz et al. managed to show that syntactic features can yield a slight improvement of 1% to 3% in precision, especially in connection with particularly ambiguous patterns.

2.2.4 Unsupervised Systems for Relation Extraction

The purpose of unsupervised learning systems for relation extraction, is the attempt to succeed in extracting relations without using any kind of labeled data. Their aim is to make human engagement in the system as little necessary as possible. Unsupervised systems are typically open relation extraction systems which are relation unspecific and do not need any predefinition of extractable relations. The idea behind this, is to make shifting to a new domain with new extractable relations less complicated and not involving any human manual intervention (comp. Banko et al. 2007). First unsupervised systems were developed, for example, by Hasegawa et al. (2004), Zhang et al. (2005) and Etzioni et al. (2004). Hasegawa et al.s (2004) system is based on the idea of extracting all kinds of relations and afterwards clustering them according to similarities of words between two entities. Zhang et al. (2005) propose a method, which also relies on clustering of entity pairs, but according to similarities of parse trees in a hierarchical cluster algorithm. This method was evaluated using the same data as Hasegawa et al. (2004) with the result of showing a better performance.

Banko et al. (2007) follow a different approach with their influential Textrunner system. The main idea of Textrunner is the training of a self-supervised learner, which is able to learn how relations are expressed in a particular language or domain. The acquired knowledge can afterwards be used to extract relations within that domain. On the basis of this idea, more unsupervised systems were created in the last years, trying to improve the algorithm, such as the systems by Eichler (2008) and Banko and Etzioni (2008). Banko and Etzioni (2008) show that many sentences expressing relations within the English language share very similar lexico-syntactical patterns, which can be grouped into eight different categories. These eight different categories, which show different frequencies in natural language text, provide an important accomplishment for open relation extraction.

Unsupervised systems have certain advantages, when used with very large corpora and in situations where the present relation types are not known beforehand. However, if the aim is to look for specified relations, they are not very adequate. If a particular set of given relations is needed, other approaches are more suitable, as, for example, weakly supervised systems.

2.3 The Linguistic Background of Relation Extraction

Within the field of relation extraction, the linguistic background should be considered in connection with the features which are used to identify relations in unstructured text. In the available literature, the linguistic background of relation extraction becomes most clear and obvious within knowledge engineering approaches for relation extraction. This makes sense, as these systems rely on rules which are immediately derived from linguistic contextual environment of the extractable relations.

However, linguistic background knowledge is taken account of more or less throughout the literature about weakly supervised and unsupervised systems, as well, depending on the respective approach. In any case, it makes sense to be aware of the linguistic environment in which relations can be detected and extracted. Also, for weakly supervised approaches, a thorough awareness of the linguistic aspects is not only useful, but necessary. Considering the linguistic environment of the particular relations carefully, can lead to taking into account linguistic features which are well suited for building a classifier with a good performance. Within this sequence of the literature review, a short overview will be given over the different feature types, which have been considered in the literature, and some ways will be pointed out, in which they have been employed in the past.

Linguistic knowledge used in relation extraction systems, can be roughly allocated within three different major levels: the lexical level, the syntactical level and the semantic level. Information on the lexical level describes lexical items or words which occur together with a certain relation. On the syntactic-structural level, the syntax of a sentence containing a relation is considered. The semantic level, finally, uses external semantic information from databases and ontologies, such as WordNet, to obtain information about semantic relationships between words or general world knowledge, which can help to discover entities and their relations. However, among the three levels of linguistic information, the majority of approaches in the field of relation extraction look at the context of a relation, either only on the lexical level or including the syntactic level.

On the lexical level, a sequence of text containing a relation can be analysed using different representations. The first and most basic lexical feature, which is considered by almost all systems, is the feature of words, which represent the entities involved in a relation. Most systems start with discovering these lexical items. Other features are derived from looking at the lexical context of a relation, or, more precisely, the words together with which a certain relation occurs. Simple systems, such as systems based on hand-crafted rules, just rely on a bag-of-word approach to look at the words which occur together with, or, in most cases, in between, the two entities involved in a relation. Agichtein and Gravano (2000), for example, use the features of a weighted bag of words. Brin et al. (1998) use regular expressions relying on a string-based lexical approach. Other systems take more lexical features into account, such as POS-tags of the words in the context.

In order to use syntactical information for relation extraction, more preprocessing is necessary. Relying on pure lexical features is still popular in weakly supervised systems, such as bootstrapping. One reason for this is that it makes them faster and lighter. Supervised systems, however, mostly make use of syntactical features, as, for example, parse-trees, because these have shown a considerable advantage in performance compared to lexical features (comp. Zhou et al., 2007). In order to be able to make use of syntactical features, chunking or parsing has to be applied. Once syntactic information has been added to the data, such as information about chunks, dependency structures or parse-trees, the system can make use of dependency-based, chunk-based or tree-based (e.g. Culotta and Sorensen, 2004) syntactic features. As mentioned above, with supervised systems the best performance can be reached by considering various syntactical representations (comp. Grishman, 2010).

Finally, using linguistic information on the semantic level means integrating semantic knowledge from ontologies into a system. Although these approaches are not as widely used as lexical or syntactic approaches, they have the advantage of including very complex information into systems, and, under certain circumstances, only require a minimal amount of training data (comp. Labsky et al., 2008). Some interesting approaches using ontologies for relation extraction can be found, for example, in the Proceedings of the first Workshop on Ontology-Based Information Extraction Systems, which was held in 2008 (Adrian et al., 2008).

2.4 Dataset: Wikipedia and Freebase

The dataset, which will be used for the project of relation extraction for people search on the web, consists of Wikipedia and Freebase. For this reason, the following section will deal with background literature about the dataset, as well as mentioning some references of relation extraction approaches which in the past made use of any of these datasources.

2.4.1 Wikipedia

After it was first launched in 2001, the web-based collaborative open internet encyclopedia Wikipedia has grown considerably, reaching 18 million articles, of which more than 3.6 articles are written in English (comp. Wikipedia, 2011). Providing such a vast amount of text written in natural language, which is easily available due to its known link structure, Wikipedia has attracted the interest of research in natural language processing.

Quite a lot of literature considering Wikipedia in the context of natural language processing has been published in the past years. Mendelyan et al. (2009) provide a comprehensive overview over the use of Wikipedia for the field of information extraction in particular and account for research, which was made in this area up to 2008. Some of the most interesting research using Wikipedia in the context of relation extraction, apart from Mintz et al. (2010), as mentioned above, has been done by Wang et al. (2007), Wu and Weld (2007), Nguyen et al. (2007a+b), Wu et al. (2008) and Weld and Wu (2010).

Wang et al. (2007) build a Positive-only relation extraction framework (PORE) relying on a support vector machine for relation classification. One of the main purpose of PORE, although it can also be used in other domains, is the population of ontologies. Wu et al. (2008) dealt with one problem connected to Wikipedia: in many cases a long tail of sparse data occurs, which is a result, for example, of incomplete Wikipedia articles, and which makes training on Wikipedia data difficult. They present three techniques to deal with this problem, namely shrinkage over a subsumption taxonomy, cleaning and augmenting the data and retrieving additional data from the web.

Wu and Weld (2007) follow an approach which is similar to the distant supervision by Mintz et al. (2010) and provide some of the basic ideas of distant supervision (comp. Mintz et al., 2010). They develop a system called Kylin, which uses the infoboxes of Wikipedia, in order to provide self-supervision for a bootstrapping method. However, Kylin is corpus-specific in so far, as it is designed to use just one single Wikipedia page at a time. Nguyen et al. (2007a) use algorithms to build core syntactic subtrees which represent the sentences containing extractable relations. This is done in order to use a tree-mining algorithm, which identifies basic elements of this semantic structure. Finally, the specific characteristics of Wikipedia are used to allocate and classify the extracted relations. In Nguyen et al. (2007b), they make use of semantic and syntactic information to form a unified structure, which can be decomposed into subsequences from which, in the following, the most frequent ones can be captured as key patterns occurring together with a particular relation. Although both papers by Nguyen at al. do not present any algorithms which show a remarkably good performance, they consider interesting ideas and methods in connection with the Wikipedia data.

Weld and Wu (2010) present WOE (Wikipedia-based open extractor), an open relation extraction system built to improve Textrunner. WOE is based on the idea of a self-supervised learner which uses unlexicalized features which are obtained by heuristically matching the data of the Wikipedia infoboxes with the corresponding text within the article. WOE shows very good performance and manages to outperform the Textrunner system by Banko et al. (2007).

2.4.2 Freebase

Freebase, which has been launched in 2007, is a collaboratively created and maintained knowledge database, which has the aim of providing a public access to the world knowledge. A short presentation of Freebase can be found by Bollacker et al. (2008). The data which is stored in Freebase has the format of tuples with more than 125 million tuples stored all together about 22 million different topics[5]. This particular format makes it easy to quickly access and use the data within applications. The main purpose of Freebase, apart from being a public repository of word knowledge, is to make the development of Web-based data-oriented applications easier (comp. Bollacker et al., 2008). As Freebase contains information that was, in fact, taken from Wikipedia, it seems very well suited to be used together with data from Wikipedia. For the field of relation extraction, for example, this is the case, as the relations which are stored in in these databases are more likely to occur within the free unstructured text of Wikipedia entries, than in any other kind of corpus (comp. Mintz et al., 2010).

2.5 Conclusions of the Literature Review for the Present Work

This literature review has looked at the field of relation extraction considering different aspects, namely the broader context of information extraction, different types of systems, linguistic knowledge connected to relation extraction and, finally and more specifically, to the context of the project the databases of Wikipedia and Freebase.

For the research topic of relation extraction for people search on the web, the weakly supervised systems are most interesting, as they do not need large amounts of hand labeled data, like supervised systems do, but still show a reasonably good performance. Another reason for choosing a weakly supervised system for relation extraction for people search, is the fact that they use a given set of relations, unlike unsupervised systems, which are open systems for relation extraction and which have no relations predefined. Looking exclusively for the attributes of people, it seems to make more sense to predefine a set of relation as a starting point for building the system. With this in mind, a focus has been put on weakly supervised systems within this literature

5 http://wiki.freebase.com/wiki/What_is_Freebase%3F

review, with a more thorough description of the system of distant supervision as a basis for the current project of relation extraction for extracting people's attributes from the web.

Moreover, it is important to consider the linguistic background of relation extraction before building a system. Linguistic knowledge, on different levels, can help in choosing powerful features which can be considered when training a classifier. Looking at the respective literature and the performance of systems using different types of features can help with this choice.

Finally, looking at the literature dealing with the dataset of Wikipedia and Freebase is important for anticipating some advantages and disadvantages, which other researchers have encountered in connection with the data set. Furthermore, it gives some insight into the potential of these databases and some inspiration for ways to apply them within the field of relation extraction.

3 The Sentence Extraction System

The topic of the present chapter is the sentence extraction system, which, based on the underlying intuition of distant supervision, will be used for extracting sentences from Wikipedia. These sentences fulfil the necessary conditions for being used as training data for a system of distant supervision. The chapter will include three main parts: first, in part 3.1, details about the used datasets will be provided. In 3.2 the methodology of the sentence extraction system will be described in detail and finally, in 3.3, a thorough manual analysis of the obtained data will show tendencies for each chosen attribute on the one hand, and on the other hand will give insights into how appropriate the mechanism of distant supervision is for the respective attribute. Based on this, possible improvements to the mechanism will be suggested.

3.1 Data Set and Chosen Attributes

Similar to Mintz et al., the dataset that has been chosen for the project of relation extraction for people search on the web, consists of data from Wikipedia and Freebase. The project uses a preprocessed Wikipedia data dump, which has, in total, a size of 78.38 GB, even though not the total amount of data will be used. The data dump consists of a set of 20 files (in the format of comma-separated-values), each 3.5 GB, which contain a preprocessed version of Wikipedia. Within these files, each Wikipedia article forms one line and each of these lines contains 3 fields, which are separated by a tab character. The first field is the title of the respective article, the second field contains the article in plain text and the third field contains an html-version of the article. As the project focuses on the attributes of people, the data is, in a first step, being filtered, so that it is possible to use only the articles about people, as will described below in chapter 3.1. The English version of Wikipedia, by its own account, is currently built up to a total of more than 3,9 million articles, with more than 2 million articles about people. This means that more than half of the articles within the English version of Wikipedia are about people.

Freebase consists of structured data which originates from Wikipedia. The fact that it is derived from Wikipedia's infoboxes, makes it very suitable for the purpose of this project, since it can be assumed that the respective entities can, without significant exception, be found in the text of Wikipedia articles.

For the present project, the choice has been made to focus on the following four attributes of people: nationality, profession, place of birth, places where a person lived. In choosing these attributes, some factors had to be considered. The most important factor is that the respective attribute is available in Freebase. Apart from that, some attributes are more suitable for this project than others. For example, the attribute of E-mail address cannot be found at all in Wikipedia articles, and the attribute of date of birth is always expressed in the same format within Wikipedia, which makes it uninteresting in the context of this project.

The part of the data of Freebase which is about people, is available on-line as a folder which contains a set of files. Included in the folder, there is a document which gives an overview over the data for a total of 2,144,989 different people, about who articles exist in Wikipedia. This file file is the starting point for the present project. It has the form of a table in tab separated values, which contains the available data for each person, subdivided into the following fields: name, id, date of birth, place of birth, nationality, religion, gender, parents, children, employment history, signature, spouses, siblings, weight kg, height meters, education, profession, quotations, places lived, ethnicity, age, notable professions, languages. One example for such a line of data can be seen below.

```
Henry Brunner   /m/02wcrxc       1838-01-22       Everton  England,United Kingdom Male     Chemist
                /m/03l1d6n
```

However, not every attribute is available for every listed person, but, instead, many fields in the table are empty. In the given example, just the fields for name, id, date of birth, place of birth, gender, profession and places lived (in the form of an identification number) are given.

As the chosen attributes for the present project are nationality, profession, place of birth and the places where a person lived, it is interesting to know, for how many people this information is available (see Table 1).

Attribute	**Number of availability in Freebase**[6]
Nationality	510 533 people
Profession	416 045 people
Place of Birth	484 435 people
Places Lived	155 158 people

Table 1: Overview over Attribute Availability

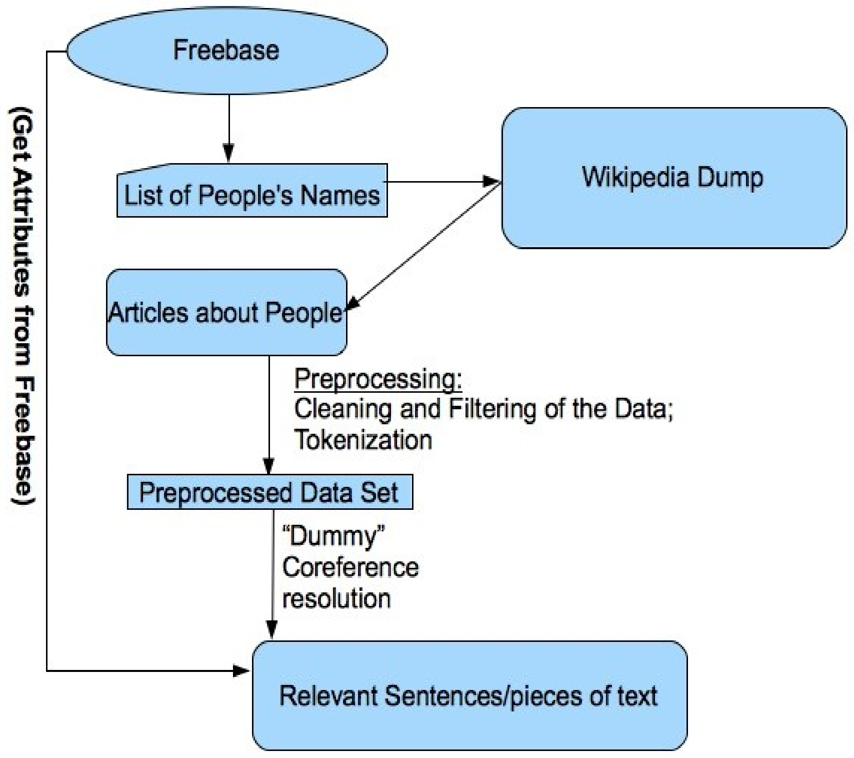
Most of these attributes are in the format of plain text within the main file and therefore directly accessible. Others consist of a specific id which has to be used to look up the actual attribute in a different file. Among the chosen attributes, this is the case for the places where a person lived, whereas the three other attributes can be found in the main file.

3.2 Methodology of the Sentence Extraction System

The methodology of the present approach of relation extraction is based on the same intuition as the research done by Mintz et al. (2010), as it has been described in section 2.2.3.1. As mentioned before, the fundamental assumption is, that in every sentence, which contains an entity pair of a person's name and a corresponding attribute, there is a high probability that this relation is expressed in some way. The attributes in focus are date of birth, profession, place of inhabitance and nationality of a person.

Due to the chosen method of weak supervision, the extraction of sentences which are later used as training data, is a crucial part of building the system. As described above, it can be said that the system is in a way supervised by the database, in this case Freebase. For this reason, the extraction of the sentences, which supposedly contain the wanted relation types, is where this kind of supervision takes place. All operations are done with the help of scripts in the programming language Python. An overview of the methodology of the sentence extraction system is depicted in a Flowchart 2.

6 Valid for December 2011



Flowchart 2: Overview over the Sentence Extraction System

The primary aim of the preprocessing step is to isolate as many sentences as necessary from the text of the Wikipedia articles. All of these sentences have to meet the condition stated by the underlying intuition of distant supervision. This condition is, given a seed of a person and an attribute which are known to be mutually connected in a wanted relation type, that the sentence contains both of them.

As the present project deals with the extraction of the attributes of people, it makes sense to only take the Wikipedia articles into account which actually have a person as topic. For this reason, the Wikipedia data is filtered in a first step. The filtering is done with the help of a list, which contains all names of people, about whom there are articles available in Wikipedia. This list is generated from the data of Freebase.

As soon as the system finds the title of an article which can be found in this list of people's names, the corresponding article is further preprocessed by just considering the plain text within the files of the Wikipedia data dump. With the plain text a tokenizer and sentence splitter is used, namely the “text sentence” package, which was implemented by Lujo (2010). “Text sentence” is a package for Python which contains a sentence splitter as well as a text tokenizer. Since the data of Wikipedia, as it is stored in the data dump, contains some irrelevant parts, as, for example, newline characters and listings of related articles and external links, the data has to be cleaned. This is done by removing single newline and tab-characters, but also by removing all sentences which contain any of the following special characters: *, |, ●, since sentences containing these symbols normally do not contain any useful information. Apart from this, sentences longer than 80 words are excluded, for the reason that these sentences are obviously far beyond the standard length of sentences in the English language. In this context, however, they can occur due to errors in tokenization or because of external links or additional information which does not belong to the actual Wikipedia article. For this reason is makes sense to exclude these sentences, as they are prone to introducing incorrect data into the system.

After the text has been preprocessed in this way, it is possible to extract the sentences which contain the name of a person as well as the corresponding attribute. For each article in this context, the name of the person, who the article is about, equals to the title of the article. The corresponding attribute is retrieved from Freebase, if it is available. Since it is possible that there are several people with the same name, all attributes for all people with the same name are retrieved. This way of handling different entities with the same name is prone to making the system slightly less accurate on the one hand, but, on the other hand, it spares a big amount of additional computational effort, which would be necessary to handle double occurrences of names in a more precise way. Nevertheless, it can generally be assumed that all sentences within a Wikipedia article about a person, are much more likely to contain the attribute for this specific person than for any other person with the same name, so that the loss of accuracy should be only minimal.

Apart from just considering the people's names, the system performs a kind of “dummy coreference resolution”. Coreference resolution refers to the task of finding a coreferential chain between sentences within a text (e.g. Mitkov, 2010). “Dummy coreference resolution”, as it is used within this project, means that the system also considers sentences containing a personal pronoun instead of a name, or just one part of the name of the respective person. This method brings about a slightly higher risk of fewer sentences actually containing the wanted relation. However, it allows to find, in total, more sentences which might express the relation. Besides, using this kind of "dummy coreference resolution" is supposed to ensure that more sentences of different formats are extracted, which, as a result, will increase the classifier's ability of correctly classifying sentences of different formats.

All these steps are reapplied repeatedly until a set of sentences has been accumulated from different articles. All extracted sentences contain the name of a person (or a pronoun in the case of "dummy coreference resolution") as well as an attribute which is known to be connected to the person in a certain relation type. As a consequence, according to the underlying intuition, most of these extracted sentences supposedly express the wanted relation type in some way.

The obtained set of sentences is then exported into a file in the format of comma separated values, containing the following fields: The extracted sentence, a label for the assumed relation type, the full name of the person, the name (or personal pronoun) as it was found in the sentence, and the attribute. This format makes it easy to access and use all the relevant information later, when the classifier is built. When using the sentences as training data for the classifier, it is assumed, even thought the data can be noisy, that all extracted sentences are positive examples for the specific kind of relation. Reaching this point, the first part of the methodology has been completed.

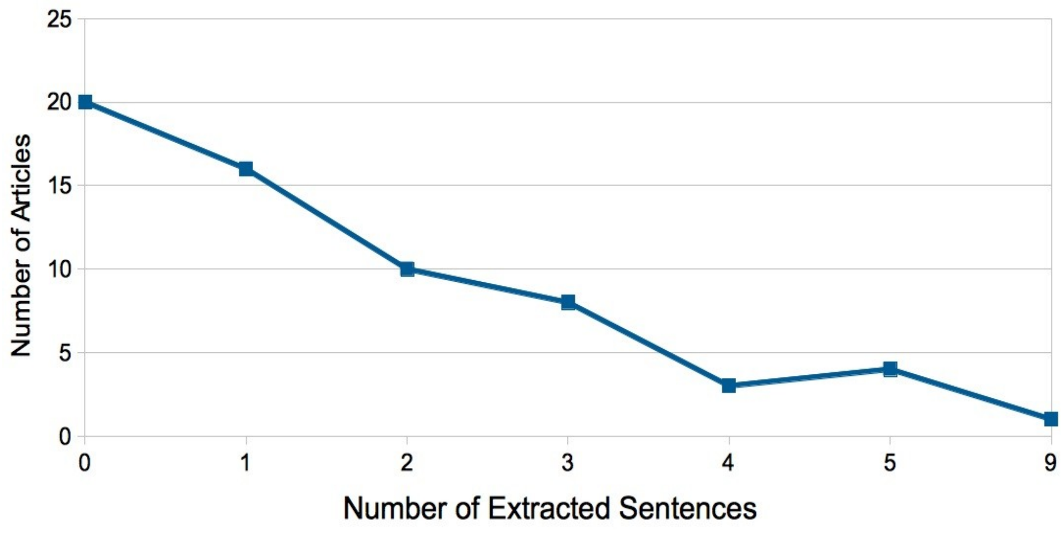
3.3 Analysis of the obtained Data for the chosen Attributes

After the methodology of the sentence extraction system has been described, this section will deal with a manual analysis of the obtained data for the four chosen attributes. For each attribute, a sample of 100 extracted sentences is taken in order to manually perform error analysis of the sentence extraction system and provide an evaluation of the method concerning in how far the underlying intuition holds true in respect to the chosen four attributes. This kind of analysis is very interesting for a weakly supervised system like the present one, as the supervision mechanism is located in the training data. The training data of the described system is provided by the extracted sentences which, according to the underlying intuition, contain entities that are known to stand in the specific relation to each other.

3.3.1 Places Lived

Typical sentences containing the relation type “places lived” are the following:

> “The fifth of six children , Coyne moved with his family from Pittsburgh's Troy Hill neighborhood to Oklahoma in early – 1961.”[7]
> (Wayne Coyne, Oklahoma)
>
> “Lenore and James had two children , Ellen and Jonathan ; they lived in New York City.”[8]
> (Lenore Marshall, New York City)
>
> “She began modeling at 15 and in 1987 graduated from Firestone High School in Akron, where she was a cheerleader and school mascot.”[9]
> (Angie Everhart, Akron)
>
> “Davis , who now lives in the rural Fentress County village of Pall Mall , also owns a construction business , Diversified Construction Co. , which builds homes , apartments and offices.”[10]
> (Davis, Pall Mall)



Graph 1: Number of extracted sentences per article
for the attribute of “places lived”

In order to extract 100 sentences containing the name of a person and a respective attribute, the sentence extraction system had to consider 143 articles about people. These were narrowed down, as only for 61 people the attribute “places lived” could be found in Freebase. Finally, the 100 sentences could be extracted from all together 41 articles. Graph 1, above, shows the distribution of number of sentences per article.

7 http://en.wikipedia.org/wiki/Wayne_Coyne
8 http://en.wikipedia.org/wiki/Lenore_Marshall
9 http://en.wikipedia.org/wiki/Angie_Everhart
10 http://en.wikipedia.org/wiki/Lincoln_Davis

The relatively high amount of articles from which no sentences could be extracted, even though an attribute could be found, can be caused by the fact that the data for "places lived" in Freebase is not very complete. For most people, just one single place is listed, even though it can be assumed that most people live or have lived in several places. Another reason can be that some articles are too short to mention places where a person lived. It is possible that an attribute is only mentioned in the Infobox as the structured part of the Wikipedia article. In other cases, it can happen that an attribute is found which actually belongs to a different person with the same name. In this case, it is not very likely to occur in the article at all and, as a result, no sentences are extracted, which, however, is not a problem for the system.

The sentences which have been extracted by the sentence extraction system for the attribute "places lived", show many different patterns, some of which will be described in the following. The different formats of the extracted sentences might be caused by the fact that the attribute "places lived" in Freebase also does not have always the same format: for most people, places where they lived are expressed as cities, but for some, places where they lived are expressed as countries. In some cases, this leads to extracting sentences, which actually contain the wanted relation, but it is not expressed through the relation between the two seed entities which have been extracted from Freebase. One example is the following sentence:

> "In 1717 he became professor in physics and astronomy in Leiden, and introduced the works of his friend Newton in the Netherlands."[11]
> (Willem's Gravesande, Netherlands)

The seed entities in this case are "Willem 's Gravesande" and "Netherlands", which are found in the sentence as "he" (pronoun from dummy coreference solution) and "Netherlands". However these two entities do not partake in a relation of the relation type "places lived". Instead, a relation can be found between "he" and "Leiden", although this relation cannot be traced back to the seed entities.

In other cases, sentences are extracted, which contain two relations, even though just one is considered. This would be the case with the sentence:

11 http://en.wikipedia.org/wiki/Willem_%27s_Gravesande

> "After graduating from Döbling Gymnasium, Drucker found few opportunities for employment in post-Habsburg Vienna, so he moved to Hamburg, Germany, first working as an apprentice at an established cotton trading company , then as a journalist, writing for Der Österreichische Volkswirt (The Austrian Economist)."[12]
> (Peter Drucker, Vienna)

"Vienna" and "Drucker" are the seeds taken from Freebase. However, it is interesting that the relation between "Drucker" and "Hamburg", in fact, is the more obvious relation of the type "places lived".

As it is the case with many important attributes of people, the places where a person lived are often mentioned at the beginning of the article. The first sentences of most Wikipedia articles contain semi-structured information, which helps the user of Wikipedia to get to the most important information about a person quicker. One example for a sentence at the beginning of a Wikipedia article would be:

> "Ali Daei (Persian:علی دایی‎, pronounced [ʔæliː dɑːjiː]; nicknamed Shahriar [ʃæhrijɑːr], meaning the King; born 21 March 1969 in Ardabil, Iran) is an Iranian retired football player and former national team coach who currently manages Rah Ahan in Iran Pro League."[13]
> (Ali Daei, Ardabil)

This sentence has been correctly extracted as it contains the name of the person, "Ali Daei" as well as one place where he lived, "Ardabil". It has, however, a format, which is very specific for Wikipedia and which differs very much from standard sentences of the English language.

Looking at 100 extracted sentences, 14 of these contain such specific format, even though all of them contain the wanted information. Including these sentences, 67 of the extracted 100 sentences contain the wanted information as opposed to 33 which do not contain the relation. However, in some cases the situation is not so clear. Consider the following sentence examples:

> "In 1948 , Richard also entered provincial politics and was elected by acclamation to the Legislative Assembly of New Brunswick as the Liberal Party member for Gloucester County."[14]
> (Ernest Richard, New Brunswick)
>
> "He studied law in Akron , Ohio, and was admitted to the bar in March 1859."[15]
> (Russell A. Alger, Akron)

12 http://en.wikipedia.org/wiki/Peter_Drucker
13 http://en.wikipedia.org/wiki/Ali_daei
14 http://en.wikipedia.org/wiki/Ernest_Richard
15 http://en.wikipedia.org/wiki/Russell_A._Alger

> “Mayawati won for the first time in the Lok Sabha elections of 1989 from Bijnor.”[16]
> (Mayawati, Bijnor)
>
> “In July 1885 , her descendants erected a tall granite memorial over her grave in what is now called the Rebecca Nurse Homestead cemetery in Danvers (formerly Salem Village), Massachusetts.”[17]
> (Rebecca Nurse, Salem Village)
>
> A key issue in the campaign was Diamondstone's opposition to Brooklyn Bridge Park, a project that Senator Connor supported.[18]
> (Martin Connor, Brooklyn)

In these examples, the relation “places lived” is expressed in some way, but it is not very explicit. Together with these sentences, some semantic issues are introduced, such as the question, whether the fact that a politician works for a party of a certain place automatically means that he or she is living there. These issues are connected to the problem of textual entailment, which deals with the question, if a certain piece of information is entailed. Considering the listed sentences, background knowledge to the nature whether a politician has to be living in the electoral ward which he represents, would have to be included for answering this question.

The same is true for sentences stating that somebody studied in a certain place. After all, it could be possible that a person lives somewhere else and comes from far away to attend a university. However, even though these kind of exceptions may exist, the probability that a student of a university or a politician within an electoral district live nearby is very high. For this reason, even though it may be prone to discussion, in general the three first sentences can be considered to contain the relation.
Looking at the fourth example sentence, however, the relation between the entities is too far to be still considered as belonging to the relation type “places lived”. The fact that somebody is buried in a place, does not necessarily mean that he or she lived in the same place while still alive. The same is true for the last example sentence. Even though, if thinking about it, the sentence is highly likely to implicate that Senator Connor works for the electoral ward of Brooklyn, it would seem rather far-fetched to assume a relation of the type “places lived” .

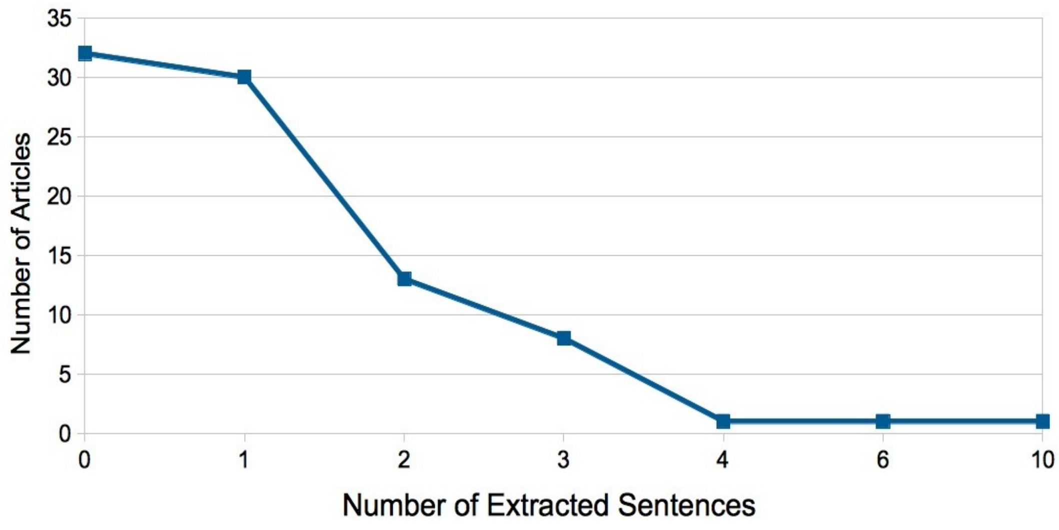
16 http://en.wikipedia.org/wiki/Mayawati
17 http://en.wikipedia.org/wiki/Rebecca_Nurse
18 http://en.wikipedia.org/wiki/Martin_Connor

3.3.2 Place of Birth

Typical sentences containing the attribute “place of birth” are the following:

> “Needham was born in Oldham.”[19]
> (Andy Needham, Oldham)
>
> “Born in Ardabil, he played for his hometown club, Esteghlal Ardabil, when he was 19.”[20]
> (Ali Daei, Ardabil)
>
> “Tisch was born in the Bensonhurst section of Brooklyn in 1926.”[21]
> (Preston Robert Tisch, Brooklyn)
>
> “Nikolaus Poda von Neuhaus (4 October 1723 – 29 April 1798) was an Austrian entomologist born in Vienna.”[22]
> (Nikolaus Poda von Neuhaus, Vienna)

In order to extract 100 sentences, the system had to consider 225 articles about people. These were narrowed down, as for only 89 of these people the attribute “place of birth” was to be found in Freebase. Finally, for only 57 of these people sentences could be extracted. For the remaining 32, even though the attribute was present, no sentences could be found. In Graph 2 the number of sentences per article can be seen.



Graph 2: Number of extracted sentences per article for the attribute of “place of birth

The number of articles about people, for whom an attribute could be found, but no sentences could be extracted, is even higher for the attribute “place of birth” than for the attribute “places lived”. This might be due to the fact that in the majority of articles, as a

19 http://en.wikipedia.org/wiki/Andy_Needham
20 http://en.wikipedia.org/wiki/Ali_daei
21 http://en.wikipedia.org/wiki/Preston_Robert_Tisch
22 http://en.wikipedia.org/wiki/Nikolaus_Poda_von_Neuhaus

standard of Wikipedia, the place of birth is included in the first sentence of the article. However, due to the format of the used Wikipedia dump, there are often remains of Wikipedia Infobox data which interfere with the beginning of the article. For this reason, it can happen that the system sees a very long sentence which is formed by the Infobox data together with the first sentence of the article and, as a result, the very long sentence is cut off by the preprocessing steps within the system, as is described above.

Another result of the the fact that the attribute "place of birth" in most cases is mentioned right at the beginning of the article, is that a high amount of sentences contain semi-structured data. Sentences containing semi-structured data are, for example, the following:

> "John Luther Adams (born January 23 , 1953 in Meridian, Mississippi) is a composer whose music is inspired by nature , especially the landscapes of Alaska where he has lived since 1978."[23]
> (John Luther Adams, Meridian)
>
> "Eleonora Anna Naria Felice de Fonseca Pimentel (Leonor da Fonseca Pimentel Chaves, Rome, 13 January 1751 - Naples, 20 August 1799) was an Italian poet and revolutionary connected with the Neapolitan revolution and subsequent short - lived Neapolitan Republic (alternately known as the Parthenopean Republic) of 1799, a sister republic of the French Republic and one of many set up in the 1790s in Europe."[24]
> (Eleonora Anna Naria Felice de Fonseca Pimentel, Rome)

The first of these two examples is a very typical example for a sentence containing "place of birth" in Wikipedia. It has a format which is shared by about one third of the sentences containing a place of birth, expressing the place of birth within brackets together with the date of birth after the name of the person. But also the second example, containing semi-structured information from other areas, as well, can be found quite frequently. Looking at the sample of 100 sentences, 42 sentences contain semi-structured information and all of them are correct in containing the relation type "place of birth". With all together 62 positive sentences within the extracted 100 sentences, sentences containing semi-structured information make up about 67% of the total of positive sentences, at least for the chosen sample.

For the attribute "place of birth" there are again some examples that might be discussable.

23 http://en.wikipedia.org/wiki/John_Luther_Adams
24 http://en.wikipedia.org/wiki/Eleonora_Fonseca_Pimentel

> "A diligent working student, he supported himself through college by working in a fastfood outlet and a video shop while studying at Holy Angel University in his hometown of Angeles City, Pampanga."[25]
> (Ronnie Liang, Angeles City)

Looking at this example, the question poses itself, whether the hometown of somebody necessarily includes the person's being born there as well. However, even though they might be discussable, these examples have been marked as negative.

Comparing with "places lived", for the attribute "place of birth" there are far less sentences that are discussable or borderline cases in containing a relation type or not. This is due to the simple fact, that from a semantic point of view, being born in a place is more specific and factual, then living in a place, in so far as it is connected to a single event with little duration: either somebody is born in a specific place or not.

Some examples among the 33 extracted sentences which do not contain the relation type "place of birth", even though they contain name and attribute, are the following:

> "A qualified veterinarian and former jockey, Weld maintains his stable , Rosewell House, in Curragh, Ireland."[26]
> (Dermot Weld, Ireland)

> "He was educated at the Berlin Academy of Art in Berlin, Germany and serves as the head of the sculptor's Department at the Avni Academy of Art in Tel Aviv as well as the acting as the head of the Israeli Sculpture Studio."[27]
> (Harry Baron, Tel Aviv)

> "He is the State Minister for Tourism and Wildlife in Uganda."[28]
> (Serapio Rukundo, Uganda)

> "At Evansville he was an All - American two–times but had to transfer due to the school dropping football as a sport."[29]
> (Sean Bennett, Evansville)

As can be seen, some of these negative examples contain country names instead of city names as their attribute, which makes them by far more unspecific and therefore more prone to not containing the wanted relation type. This is an incoherency which is introduced by the data of Freebase.

25 http://en.wikipedia.org/wiki/Ronnie_Liang
26 http://en.wikipedia.org/wiki/Dermot_Weld
27 Article was deleted in present Wikipedia.
28 http://en.wikipedia.org/wiki/Serapio_Rukundo
29 http://en.wikipedia.org/wiki/Sean_Bennett

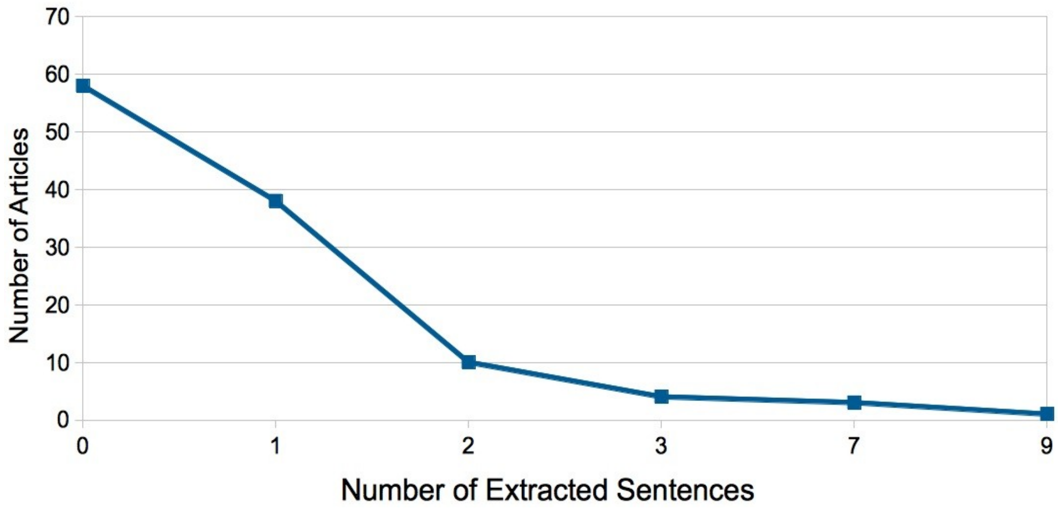
Furthermore, it is interesting to note that, even if the attribute is in the format of a city name, for some people many sentences can be extracted, with all or at least most of them being negative. This is due to the semantic reason that there are people who are very active in the same place they were born in. Graph 2 shows that there is one article with 10 extracted sentences, interestingly all of them negative. From this fact it can be concluded, that it could make sense to just consider the first 2 or 3 sentences in an article for extracting the sentences which might contain a relation of the type "place of birth". However, it has to be said that one important characteristic of sentences containing the relation of "place of birth" consists in the fast that they contain the word "born", without exception. For this reason, a rule-based system is likely to reach better results for this specific attribute than a machine learning system which works with the described mechanism of distant supervision.

3.3.3 Profession

Typical sentences containing the attribute "profession" are the following:

> "Peutz (7 April 1896 - 24 October 1974) was a Dutch architect."[30]
> (Frits Peutz, architect)
>
> "Though his employers were sometimes reluctant to hire him knowing that he was blind, his reputation grew as it became apparent that he was a capable mathematician and teacher."[31]
> (Abraham Nemeth, mathematician)
>
> "On November 12 , 2008 Lemonis signed a contract for rest of the 2008 - 09 season and replaced Ewald Lienen as the head coach of Panionios f."[32]
> (Takis Lemonis, coach)
>
> "He studied human medicine at the Charite Universitätsmedizin in Berlin from 2000 to 2007 and now works as an assistant surgeon."[33]
> (Colin Grzanna, surgeon)

30 http://en.wikipedia.org/wiki/Frits_Peutz
31 http://en.wikipedia.org/wiki/Abraham_Nemeth
32 http://en.wikipedia.org/wiki/Takis_Lemonis
33 http://en.wikipedia.org/wiki/Colin_Grzanna



Graph 3: Number of extracted sentences per article
for the attribute of "profession"

In order to extract 100 sentences, the system had to go through articles about 326 people. These were narrowed down to 113, as this is the number of people among these, for who the attribute "profession" could be found in Freebase. Finally, the system managed to extract sentences from 55 of the articles about these people. For 58, even though the attribute "profession" was found in Wikipedia, the system was not able to find sentences containing the name of the person together with this attribute. Graph 3, gives an overview over how many sentences could be extracted per article.

The high number of 58 articles for people for whom the system did not manage to extract sentences, even though the attribute "profession" was present in Freebase, is probably due to some discrepancies in the data of Freebase as compared to Wikipedia. For example, looking at the data for the Welsh singer Gruff Rhys, the retrieved attribute for profession from Freebase is "singer". However in Wikipedia, his profession is expressed in the following way:

> "Gruffydd Maredudd Bowen Rhys (Welsh pronunciation: [ˈɡrɨ̞fɨ̞ð maˈrɛdɨ̞ð ˈbowɛn ˈr̥ɨːs]; born 18 July 1970 in Haverfordwest) is a Welsh musician, performing solo and with several bands, including Super Furry Animals who obtained mainstream success in the 1990s."[34]
> (Gruff Rhys, singer)

It can be seen, that within the Wikipedia article, the profession of "singer" is encoded as "musician, performing solo and with several bands" and therefore cannot be found by the system. "Singer" on the other hand does not appear in the text of the Wikipedia article.

34 http://en.wikipedia.org/wiki/Gruff_Rhys

A similar issue exists with the following sentence:

> "Marion Barton Skaggs (April 5, 1888, Missouri - May 8, 1976, Alameda County, California) (nicknamed M.B.) was an American businessman and leading member of the Skaggs Family of retailers who expanded the predecessor of Safeway into a major supermarket chain."[35]
> (Marion Barton Skaggs, businessperson)

The retrieved attribute from Freebase is "businessperson", however in the text of the Wikipedia article it is expressed as "businessman". This leads over to an important issue concerning Freebase data for the attribute "profession" and another reason, why the system only managed to extract relatively few sentences in regard to the amount of people for who the attribute was present: in Freebase occupation titles are not encoded in a gender specific way. This means, even though an article can be about a women who works as an actress, the attribute which is retrieved from Freebase is "actor". Besides the fact that this makes the system miss sentences which should be extracted, it causes further errors, as for example extracting sentences that should not be extracted:

> "Zellweger has been dating actor Bradley Cooper since 2009."[36]
> (Renée Zellweger, actor)

This error would not have happened if the system had been looking for "actress" instead of "actor". In future research it would be preferable to find a way to cope with this kind of noise in the data, caused by gender unspecific occupation titles.

As it is the case with the attribute of "places lived" described above, there are also some sentences for the attribute "profession", for which it is not so clear if they contain the wanted relation type or not.

> "At the 1918 general election, he did not stand in Newton (which was won by the Labour Party politician, Robert Young), but was elected to the newly - formed constituency of Aldershot that year."[37]
> (Roundell Palmer, 3rd Earl of Selborne, politician)
>
> "Prus' stories, which met with great acclaim, owed much to the literary influence of Polish novelist Józef Ignacy Kraszewski and, among English - language writers, to Charles Dickens and Mark Twain."[38]
> (Bolesław Prus, novelist)

35 http://en.wikipedia.org/wiki/Marion_Barton_Skaggs
36 http://en.wikipedia.org/wiki/Ren%C3%A9e_Zellweger
37 http://en.wikipedia.org/wiki/Roundell_Palmer,_3rd_Earl_of_Selborne
38 http://en.wikipedia.org/wiki/Boles%C5%82aw_Prus

Even though in these sentences, the wanted relation type is in some way stated implicitly, there is no direct connection of this relation type between the seeds of name of the person and attribute in the sentence. The attributes in both cases are, in fact, connected to other people. For this reason, these sentences have to be marked as not containing the wanted relation type.

With 85 sentences, which, in fact, contain the wanted relation type "profession", as opposed to 16 sentences which do not contain it among the 100 sentences of the chosen sample, it can be said that the underlying intuition reached a comparatively good result for this relation type. The main reasons for this are the semantic restrictions that come along with occupation titles in general: when occupation titles occur together with a person in a sentence, in the majority of cases, a relation of the type "profession" holds true between the person and the title. It can be explained by the fact that the concept of an occupation title normally already includes that it refers to a person. This is especially true for Wikipedia articles, as they provide a description of a person, which mainly mentions facts about the person, with the relation type "profession" being the most obvious relation that a person can have to an occupation title.

Considering the three articles from which more than five sentences could be extracted (see Graph 3 above), it is interesting to note that two of them are from the area of sports and include the occupation title "coach". Even though, taking a closer look at the extracted sentences from these articles, most of them, in fact, contain the wanted relation type, it would make sense to limit the number of sentences taken from each article. A prediction can be made that a majority of articles which contain more than 5 sentences with the name of the person together with the respective occupation title are from the area of sports. Limiting the number of sentences, thus, would make a bias towards this thematic area less prominent.

3.3.4 Nationality

The attribute of nationality, as compared to the other three examined attributes, poses a special case in some ways. In Freebase nationalities are encoded as simple names of countries, as for example:

(Louis Pio, Denmark)
(Giulio Tremonti, Italy)

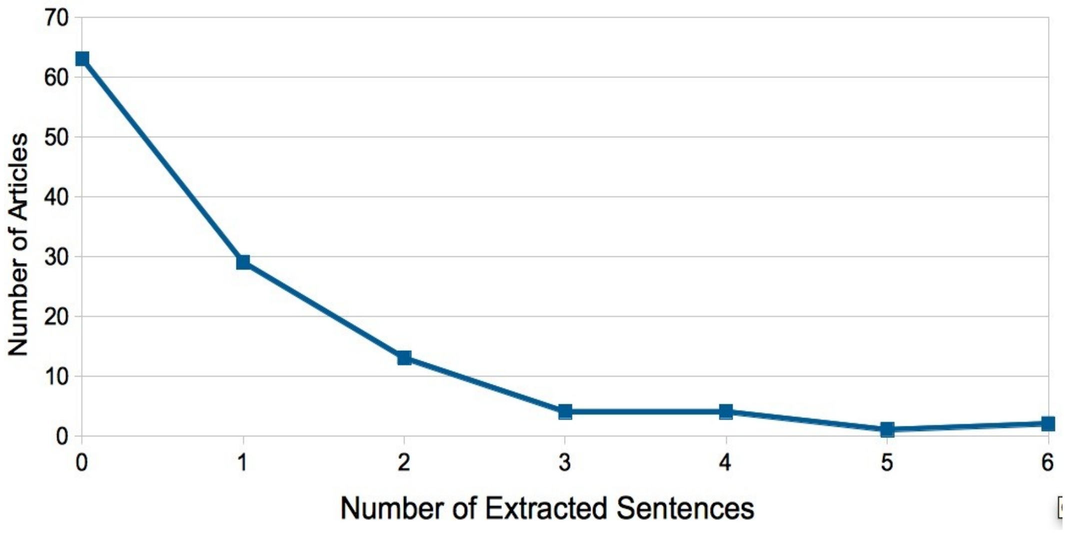
This makes the quote of sentences containing the wanted relation type of "nationality" among those that have been extracted very low, as nationalities, in most cases are encoded by the use of adjectives. Compare the following two sentences:

> Kuljanin (born April 2, 1985 in Sarajevo) is a male basketball player from Canada, who plays as a center.[39]
> (Vladimir Kuljanin, Canada)
>
> Octave Crémazie (April 16, 1827 – January 16, 1879) was a French Canadian poet.[40]
> (Octave Crémazie, Canada)

Looking at several sentences from Wikipedia articles which contain the relation type of "nationality", it is obvious that the second type, expressing the nationality through an adjective, is by far more frequent than the first one.
For this reason, a list of the different formats in which nationalities can be encoded is taken from Wiktionary[41] which makes it possible to additionally extract sentences in which a nationality is expressed in a different way. In the following, the obtained data of both types of sentence extraction system will be described and discussed.

The system which just considers country names in the format as they have been retrieved from Freebase needs to go through articles of 239 people in order to extract 100 sentences containing name and attribute. These were narrowed down to 114, as for 125 people the attribute was not present in Freebase. Finally, from 51 of these 114 articles sentences could be extracted. An overview of how many sentences could be extracted per article is given in Graph 4.

39 http://en.wikipedia.org/wiki/Vladimir_Kuljanin
40 http://en.wikipedia.org/wiki/Octave_Cr%C3%A9mazie
41 http://en.wiktionary.org/wiki/Appendix:Countries_of_the_world



Graph 4: Number of extracted sentences per article for the attribute of “nationality” (only country names)

The high amount of 63 people for whom no sentences could be found, even though the attribute “nationality” was present in Freebase, shows some evidence that it was not very easy for the system to find sentences where the name of the person occurred together with the name of the country of which they have the nationality.

Looking at the sentences which were extracted by this system, consider the following:

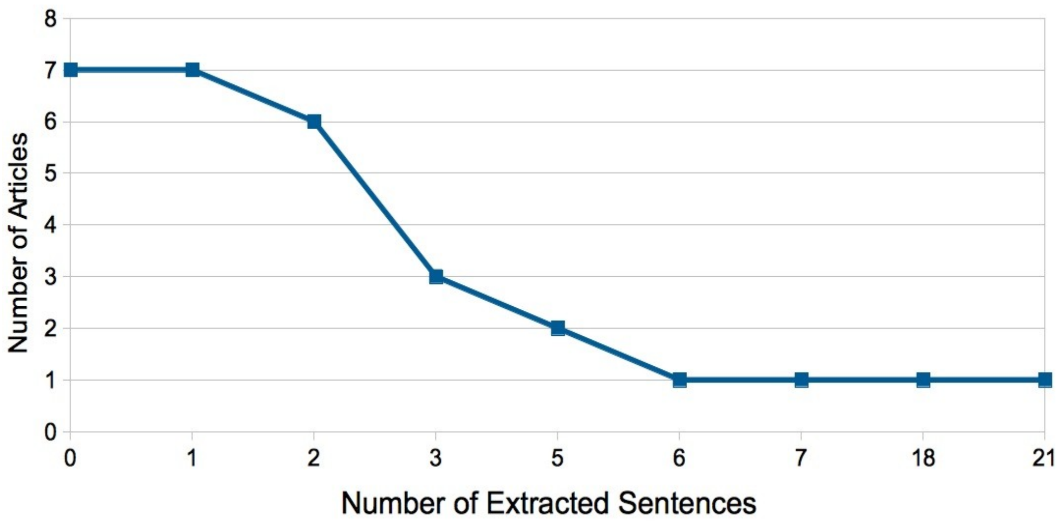
> “He was born in Denmark.”[42]
> (Jørgen Reensberg, Denmark)
>
> “He is also a member of the Mexico national basketball team.”[43]
> (Karim Malpica, Mexico)
>
> “Ryan represented Australia at Expo '74 in Spokane, Washington, USA, with Judy Stone and Rolf Harris.”[44]
> (Ross Ryan, Australia)
>
> “Giusep Nay (born August 9, 1942 in Trun, Grisons) was the president of the Federal Supreme Court of Switzerland for the years 2005 and 2006.”[45]
> (Giusep Nay, Switzerland)

The information which can be found in each of these sentences can provide a hint that a person has a certain nationality, but this nationality is not expressed explicitly: semantic background knowledge is needed to make the conclusion that a relation of the type “nationality” is present.

42 http://en.wikipedia.org/wiki/J%C3%B8rgen_Reenberg
43 http://en.wikipedia.org/wiki/Karim_Malpica
44 http://en.wikipedia.org/wiki/Ross_Ryan
45 http://en.wikipedia.org/wiki/Giusep_Nay

Within the sentences extracted by the system, there are many sentences of this kind, most of them stating that a person played for the national team of a country. If they are counted to be positive in containing the wanted relation type, 46 positive sentences as opposed to 54 negative sentences can be counted for the chosen sample of 100 sentences. However, if these sentences, which only implicitly state the relation type, are not taken into account, only 9 sentences which explicitly state the relation type can be counted, as opposed to 91 which only implicitly state or do not state the relation type. With only 9% of sentences actually containing the wanted relation (at least for the examined sample of 100 sentences), the data for the attribute "nationality", using the seeds from Freebase without any modification, is very noisy. A system trained on this kind of data is not very likely to show a good performance in classifying new sentences according to whether they contain the relation type of "nationality".

The system which uses different formats retrieves the nationality in the form of the name of the country (or countries for people with more than one nationality) from Freebase and subsequently retrieves all other formats. This means that, for example, if the nationality in Freebase is "Ireland", the system looks for any of the following: "Ireland", "Irishman", "Irishwoman", "Irish". If any of these words can be found in a sentence together with a name or a personal pronoun, the sentence is extracted by the system.



Graph 5:
Number of extracted sentences per article for the attribute "nationality"(different formats of nationalities)

Considering all formats, the system has to go through articles about 56 people in order to extract 100 sentences. These 56 articles are narrowed down to 33, as for 23 of these 56 people, the attribute nationality could not be retrieved from Freebase. As the system could not find attributes for 7 people, the 100 sample sentences are finally extracted from altogether 49 articles.

The fact that the system needs to look through just 49 articles until a total of 100 sentences have been collected, is due to the fact that it succeeded to extract comparably many sentences per article. For two of the articles from which sentences have been extracted, even 18 and 21 sentences which meet the conditions could be found. The explanation for these high numbers is that the expression of nationalities in all formats is very unspecific and can refer to many kinds of relation types apart from that of "nationality". If a person was very much connected to a country in a special way (e.g. as a national hero or politician), it can happen that the name of the country and the nationality appear very frequently in an article. Consider for example the following sentences which have all been extracted from the same article:

> "In this <u>Japanese</u> name , the family name is <u>Abe</u>."
>
> "While still a student, <u>he</u> volunteered for military service during the First Sino - <u>Japanese</u> War."
>
> "After the war, <u>Abe</u> graduated from the Imperial <u>Japanese</u> Army Academy followed by the 19th class of the Army War College."
>
> "During his short four month tenure, <u>Abe</u> sought to quickly end the Second Sino - <u>Japanese</u> War, and to maintain japan's neutrality in the growing European conflict."
>
> "After his return to <u>Japan</u>, <u>Abe</u> joined the House of Peers in 1942 , and accepted the largely ceremonial position as president of the Imperial Rule Assistance Political Association."[46]
> (Noboyuki Abe, Japan)

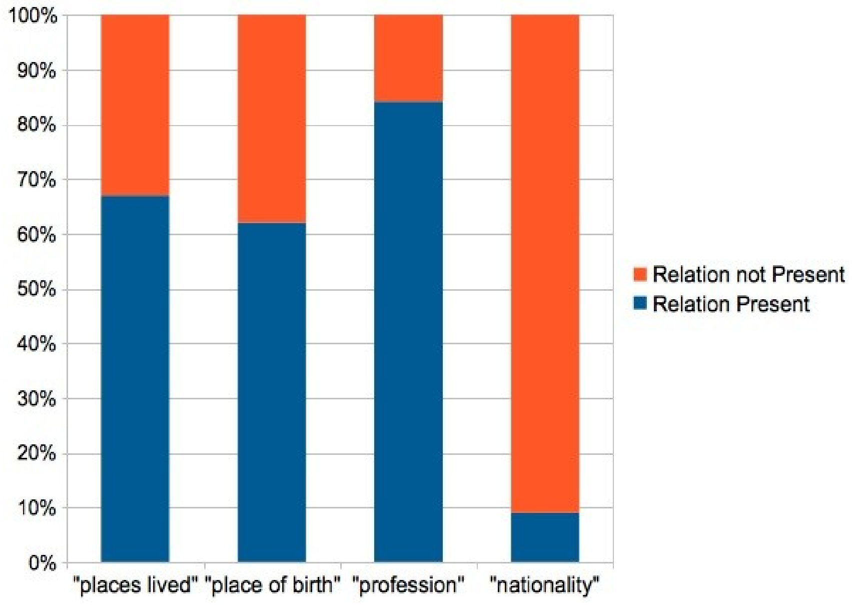
None of these sentences actually expresses the wanted relation type, at least not in an explicit way. However, looking at the articles from which more than one sentence could be extracted, it becomes clear that in most cases, if the relation type "nationality" is expresses, it normally happens in the beginning of the article. With this in mind it makes sense to limit the extracted sentences to the first 3 sentences of the article. Not only does this avoid the extraction of many sentences, which are to a high degree less likely to contain the wanted relation type, but also, the number of sentences from the same area

46 http://en.wikipedia.org/wiki/Nobuyuki_Abe

(sports, politics, authors, etc.), which possibly introduce a bias to this area, can be reduced.

3.3.5 Conclusions of the Manual Analysis

Even though the examined sample for each of the chosen attributes with 100 sentences is relatively small, the manual analysis shows some tendencies of the extracted data. How many sentences containing a wanted relation type could be extracted based on the underlying intuition, is very different from attribute from attribute. Graph 6 shows an overview over the number of sentences which are identified as containing the relation, out of a sample of 100.



Graph 6: Percentage of Sentences Containing Relation Type

In general, it can be concluded that how well the underlying intuition works for each attribute mainly depends on semantic factors. One of these semantic factors is, for example, how semantically specific a word that encodes an attribute is for a wanted relation and which concept it encodes, as has been explained above.

While the underlying intuition worked relatively well for the attribute of “profession”, with 85 of 100 sentences containing the wanted relation, the extracted data for the attribute of “nationality” is very noisy with, strictly speaking, only 9 of 100 sentences containing the wanted relation type. As it cannot be expected that a system, trained on this kind of very noisy data, will show a good performance, some improvements for making the data less noisy have been proposed for the attribute of nationality. A general improvement, which can reduce noise in the data for all examined attributes, is limiting the sentences, extracted from a single article, to the first three sentences that can be found by the system. Like this, not only a bias towards certain areas (e.g. sports or politics) can be reduced, but also it has been discovered that sentences at the beginning of an article are more likely to contain a relation type of the four examined ones, than sentences which occur towards the end of an article.

For some attributes, even though the underlying intuition holds true to a satisfactory degree, it is to be expected that a rule-based system still will perform far better than a system based on this kind of distant supervision. The attribute for which this is the case among the examined attributes, is “place of birth”, as the word “born” is an important indicator for this relation type and present in all sentences which express this relation.

One issue, which is present in different degrees for all the examined attributes, is that of semi-structured information: In the first sentences of a Wikipedia article about a person, a condensed overview is given over the most important facts about this person. These semi-structured sentences are different from standard sentences of the English language in so far as they contain a more structured format, which is achieved by the use of brackets and other means of structuring. Semi-structured information as a feature of Wikipedia might be user friendly, as it allows the user to get to the most important information faster, but for the current research it is an issue in so far as these sentences have a format which is very specific for Wikipedia. A system trained with this kind of Wikipedia specific data can be expected to work well for extracting new information from Wikipedia, but if used with other text types it will probably perform far worse.

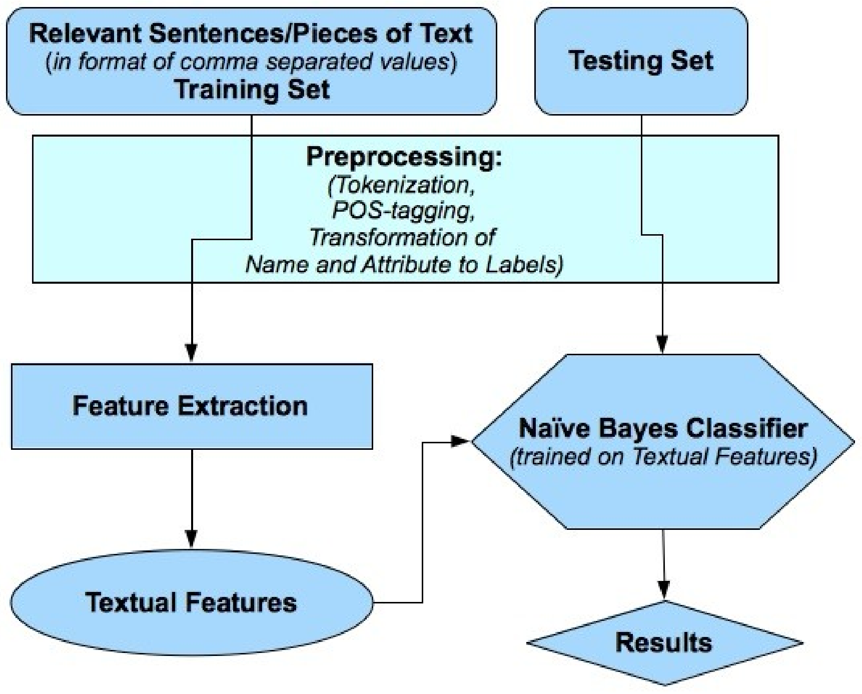
Looking at the sample of extracted sentences for all attributes, it can be seen that the “dummy coreference resolution” performs well, at least in terms of precision. The majority of the sentences which have been extracted through “dummy coreference resolution”, in fact, refer to the person that the article is about. The reason for this, however, simply depends on the special characteristics of Wikipedia articles, as parts of an encyclopedia: the majority of the sentences within an article refer to the person the article is about.

4 Methodology of the Machine Learning System

The methodology of the second part of present project consists in building an example system of machine learning which uses the sentences which have been extracted by the sentence extraction system as training data. The example system focuses on the attribute of "nationality" and is supposed to show the impact of changes within the training data, in order to make it less noisy, on the performance of the classifier. The input to the system are the files in the format of comma separated values (csv) which have been generated by the sentence extraction system, each containing 2000 sentences, one with supposedly positive examples and one with supposedly negative examples for the relation nationality. The mechanism for extracting the positive sentences has been previously described in chapter 3, whereas the mechanism for extracting the negative sentences will be presented in part 3 of the present chapter.

The sentences are preprocessed and features are extracted, which will be described in detail below. With the extracted features, a classifier is trained which can then be used to classify sentences according to if they contain the relation "nationality" or not. An overview of the whole procedure is given in Flowchart 3.

For training the classifier, the natural language toolkit for Python (NLTK) is used.



Flowchart 3: Overview over the Machine Learning System

4.1 NLTK: the Natural Language Toolkit for Python

The natural language processing toolkit (NLTK) is a collection of libraries and programs for processing natural language in Python. The initial release of NLTK took place in 2001 and since then the project has been led by Steven Bird. NLTK has the big advantage that it is open-source and very well documented, on-line, as well as in the book “Natural Language Processing with Python” (Bird et al., 2009). One reason why NLTK has been chosen for this project, is the fact that it works within Python. This means that, theoretically, it would be easy to combine both systems, the sentence extraction system and the classifier system, to one united system which is able to provide the training data and train the classifier. Another reason is, that it contains tools for preprocessing, especially a pos-tagger, which is used for tagging the sentences of the input data.

Using NLTK, the preprocessing of the training data, the extraction of the features and, finally, building the classifier can all be done by the same system. The Naïve Bayes classifier which is part of NLTK has been chosen for the machine learning approach within this project, for the main reason that a good documentation is provided which makes its use relatively easy and understandable. Additionally, within the Python script, all steps that are taken in order to extract different features from the training data and use them to train the classifier become very transparent. Furthermore, including everything in one script makes it easy to add and modify features as it is needed during the development process, as well as to include training and testing data from different sources.

4.2 Naïve Bayes Classifier

The Naïve Bayes Classifier is chosen for this project for the simple reason, that it is included in the NLTK package and even though compared to other classifiers it is relatively simple, it still can show a good performance. Mintz et al. train a logistic regression classifier which uses conjunctive features which, in order to find matching features, they need to have all their conjuncts in common. As Mintz et al. point out, this methodology makes sense with very large amounts of data. Hence, they use the data of

800,000 Wikipedia articles for training and of 400,000 articles for testing, as mentioned above in the literature review. However, due to restrictions in computational power, a smaller sample will be used within the present project.

Being more basic in its concept and not using conjunctive features, the classifier of the present project is expected not to be able to reach up to the performance of the classifier described by Mintz et al.. However, the primary aim of the described example system for the attribute "nationality" is to compare the performance of the system, using different kinds of training data, against each other in order to get an idea of the impact of the different training data on the system. For this purpose, the Naïve Bayes classifier seems adequate. A good description of Naïve Bayes classifiers is provided for example by Bird et al. (2009),

A Naïve Bayes classifier assumes that all features should be treated independently to each other and when making a decision of classification, all features are equally important and decisive. The assumption that each feature is independent from all other features, in most cases, can be described as somewhat unrealistic "naïve", as most features are very likely to depend on other features in some way. However, this assumption simplifies the classifier considerably.

Naïve Bayes classifiers are based on the Bayes Theorem:

$$P(T/X)=\frac{P(X/T)P(T)}{P(X)}$$

In order to classify an new, unseen example, the first step consists in calculating the prior probability which depends on which class label is the most frequent in the training set. The second step combines this prior probability with each feature. This makes a prediction possible concerning how the new example should be labeled. In the following, a likelihood estimate is reached for each class label by combining this prior probability with the effect that each independent feature has. When the highest likelihood estimate is found, the corresponding class label is given to the new example. Smoothing becomes necessary if there is a feature which never occurs together with one of the class labels, because it would mean that the probability for a label, given a feature equals to zero with the consequence that the label is never given to a new unseen example. A remedy for this is to add 0.5 to every value, so that zero never occurs. The

Naïve Bayes classifier, which is integrated in NLTK and used for the present project, uses this smoothing technique among others.

4.3 Data for Training and Testing

For training the classifier, positive as well as negative training data is necessary. The positive training data originates from the sentence extraction system as it is explained above, in chapter 3. In order to show the impact of the modified intuition on the performance of the machine learning system, two positive training sets are provided for the attribute of nationality: one training set consists of the extracted sentences that only consider the attribute of "nationality" in the format of country names, as they were present in Freebase. This is the original mechanism of distant supervision. The other one takes two proposed changes on the sentence extraction system for the attribute of "nationality" into account. This means that different formats in which nationalities are expressed, are considered. Additionally, the sentence extraction system only extracts the first three sentences that can be found within an article.

In order to achieve a negative training set for their system of distant supervision, Mintz et al. used sentences containing randomly selected entity pairs for which no kind of relation is listed in Freebase. For the approach of the present project, a similar method, which is however slightly closer to the underlying intuition, is chosen: the negative training set is provided by a slightly modified form of the sentence extraction system. The sentence extraction system looks through articles about people and extracts all sentences containing the name of the person, the article is about, occurring together with a nationality which is not connected to that person through the attribute of "nationality" in Freebase. This is done with the help of a list, which contains all nationalities minus the nationality for the specific person the article is about. As an argumentum e contrario, derived from the underlying intuition of distant supervision, this method is expected to yield sentences which do not contain the relation type of "nationality". The negative training set is adapted to the respective positive training set: for the first training set, the negative sentence extraction system extracts sentences only containing country names and for the second training set, it extracts sentences containing nationalities in all formats. This has been done in order to achieve a better comparison of the original method of distant supervision to the modified version. The negative data, like previously the positive data, is stored in files with comma separated values, containing the same fields as the files for the positive data.

Since the data, provided by the sentence extraction system, is potentially noisy, it is not very suitable for precise testing. For this reason, a testing corpus is built manually, that contains 1000 sentences: 200 sentences are taken from the first sentence extraction (nationalities only as names of countries) system, 200 are taken from the second sentence extraction system (nationalities in all formats) and 300 sentences are taken from each of the two negative systems. Only the positive sentences are manually marked as positive, as it is expected that with the negative sentence extraction system, unless due to inconsistencies in Freebase, all sentences should in fact be negative for the relation type of nationality. It has been decided to use 100 more sentences of negative data than of positive data for two reasons: firstly, because negative sentences are expected to be less noisy and secondly, because using more negative examples the data will correspond better to the prior probability assumption of the Naïve Bayes classifier, as explained in section 4.2 above.

4.4 Preprocessing of Data and Selection of Features

The system is supplied with data in the format of a comma-separated-values, containing four fields which are separated by commas: the sentence extracted by the sentence extraction system, a label of the relation type, the full name of the person, the name of the person as it could be found in the sentence, and the attribute in the format in which it could be found in the sentence. This format makes it easy to convert the data into a tuple, containing these four fields. All together three files form the input to the system: One file with 2000, potentially noisy, positive sentences for the relation type "nationality", one file with 3000 negative sentences for the relation type "nationality", and one file with testing data which was prepared as described above. However, later for training not all of this data will be used, for the reason that the Naïve Bayes classifier of NLTK becomes very slow with an input of more than 1000 sentences in total. Due to restrictions of this kind, the data had to be cut to smaller amounts.

Once the data has been transformed into tuples, the sentences are once more preprocessed through tokenization, part-of-speech tagging, bigram tagging and substitution of the entities which partake in the relation by labels. For every alteration that is done, the old format of the sentence is still kept in the tuple. For tokenization the text-sentence module for Python which has been described already in connection with the preprocessing steps for the sentence extraction system, is used one more time. After tokenization, part of speech tags are added to the sentence by using the standard part of speech tagger from NLTK. This part of speech tagger, which is part of the NLTK package, is a maxent part of speech tagger which was trained on the Penn Treebank corpus. For bigram tagging an n-gram tagger which is also part of the NLTK package is applied. From findings in the manual analysis, it has been concluded that the removal of stopwords does not make any sense in this context: within the extracted sentences stopwords often contain important information regarding if a relation holds between two entities or not. For this reason, instead of removing stopwords, just articles, punctuation and the previously substituted class labels are being removed.

Concerning the selection of features, there are three areas to which features can apply. The whole sentence, the sequence of words between the entities, the sequence of words before the first entity and the sequence of words behind the second entity. Three different kinds of lexical features are taken into account: bag-of-word features, bigram features and features of part-of-speech tags. All of these features are binary features. The following table (Table 2) shows an overview over which features are extracted from which area of the sentence.

Features	Whole Sentence	Between Entities	Before 1st Entity	After 2nd Entity
Bag-of-Words	Yes	Yes	–	–
Bigrams	Yes	Yes	–	–
Part-of-Speech	–	–	–	–
Part-of-Speech Bigrams	–	Yes	Yes	Yes

Table 2: Selection of Features for the ML system for the attribute of “nationality”

The choice of features has been made in correspondence to findings in the manual analysis. The area between the two entities, in this regard, seems to be most important for relation-specific patterns. For this reason, most features are extracted from this area. For the area before the first entity, a window of four words is taken into account. The same window size also applies for the area after the second entity.

During test runs of the classification system, it turned out that part-of-speech unigrams are not specific enough to be distinctive in this context. For this reason, finally, they have not been considered for the selected features. Bag-of-Word features uses a list of the one thousand words with the biggest occurrence in the test set and bigram features uses a list of the 200 most frequent bigrams in order to avoid sparse data.

Many approaches, among them Mintz et al. use the feature of a flag, including information concerning which of two entities within a relation occurs first in the sentence. However, as during manual analysis, no evidence has been found for the importance of this feature, it has been decided to not take it into account.

Even though Mintz et al. (2009) suggested that syntactic features can enhance performance of the classifier, as mentioned above, they are not taken into account for the present research. The use of parsers, which would be necessary in order to use syntactic features requires a high amount of additional computational power. As the object of the present system is mainly to show tendencies of the data, these additional computational expenses are not necessary. Nevertheless, for future research it can be interesting to examine the impact of changes within the training data regarding syntactic features as well.

4.5 Results and Evaluation

The evaluation of the classification system is done in terms accuracy. Accuracy refers to how well the classifier is able to reproduce the labels in the previously labelled test set. This kind of evaluation is possible, as a testing set was predefined. In other, real life, scenarios of distant supervision, it is hard to measure accuracy, as the classification system works with data which has not been previously labelled. This data could for example be unstructured text from the internet, from which relations are extracted. In these scenarios it is only possible to do evaluation in terms of precision. Precision refers to the number of correctly classified new instances among the number of total classified new instances. Mintz et al. used two methods for measuring precision, as mentioned above: automatically, by using relation data, which was not previously used for training and manually, deciding for each sentence marked as positive, if the relation in fact holds true.

However, the first method seems questionable, as, even though a sentence can contain two entities for which partake in a specific relation listed in Freebase, this does not necessarily mean that the sentence actually states this specific relation. Various counterexamples have been shown in chapter 3 of the present work, especially for the attribute of "nationality".

As mentioned above, the NLTK Naïve Bayes classifier becomes very slow under a load of around 1000 sentences in total. For this reason, the decision had to be made to use smaller samples for testing. From the total of 2000 positive and 2000 negative sentence which are provided, random samples of 250 for each positive and negative training are taken. From the 1000 sentences of testing data, which has been provided, random samples of 300 sentences are taken for each run, so that the system does not have to cope with more than 800 sentences at a time. By taking different random samples for each run, a bigger variety in the data is reached. For each of the two different kinds of training data, six runs are preformed with the results shown in Table 3.

	Original	**Modified**
Run 1	60	76
Run 2	45	62
Run 3	61	65
Run 4	60	61
Run 5	62	72
Run 6	57	67
Average	57.5	67.1

Table 3: Results of six runs of the system for "nationality" (measured in accuracy)

The relatively high results that the system reaches, are probably due to the semi-structured data, as in the negative training set a by far lower amount of semi-structured data is to be expected, which makes semi-structured data characteristic for positive examples. Regardless of this, comparing the results of the variations of the system, using different training data, even though the differences in performance are only slight, a certain tendency can be seen in favour of the modified form of the sentence extraction system. However, it has to be kept in mind that these results were reached within a very restricted environment and with very little data. For this reason it is necessary to investigate further in future research whether and in how far, in a different environment, the proposed changes on the training data can enhance the performance of a system of distant supervision.

After training the classifier on the training data, the system gives an overview over the 20 most informative features within the training set. Within these most informative features a bias to specific thematic areas can be seen. It is striking that within the most informative features of the system trained on the data from the original, unmodified sentence extraction system, the bias towards the area of sports is remarkably higher than in the modified relation extraction system. Some of the highest ranked most informative features for giving a positive label to an unseen examples, is if they contain one of the words "football", "player" or "won". As this is not the case for the modified system, it can be concluded that there has already been a success in reducing the bias towards this thematic area within the modified system. Another interesting fact is that among the most informative features for the system trained on the data from the modified version of the sentence extraction system, a by far bigger amount of part-of-speech tag features can be found. An explanation for this can be that with the modified training data more common structures can be found, rather than common words.

4.6 Possible Improvements of the System

The most obvious improvement to the described system, is to use more data in total. Results with around 5000 sentences for each system can be quite different from the achieved results within the present project. Furthermore, testing should be one on a bigger annotated corpus, or, respectively, measuring precision through manual analysis of the newly labelled testing set as the output of the system. Finding a method for evaluation which does not involve any human involvement would be even better but so far these methods do not yet meet the standards of human evaluation, for reasons discussed above. Furthermore, evaluation with data from a different source than Wikipedia can be very interesting in order to see how Wikipedia specific a system is which was trained on Wikipedia data.

Another improvement to the system, which can be even more important when processing different relation types at the same time, would be the adding of a confidence score for each classified sentence or pair of entities, as was done in the system by Mintz et al..

Although the Naïve Bayes classifier, which was applied, served well in getting ideas of tendencies within the data, it is very much likely that other classifiers, such as logistic regression, as used by Mintz et al. can reach a better performance with this specific training set, for example by combining features. It would be interesting to compare the performance of different classifiers against each other using data extracted based on the underlying intuition and modified underlying intuition.

Apart from improvement in terms of data, the system could also be improved by testing the impact of different additional features. Some features that could be interesting are for example the following: the number of words between the entities, the length of the sequences before the first and after the second entity within the sentence and 3-gram features, using an effective backoff tagger in order to avoid sparse data. Furthermore, as already mentioned above, syntactic features could be applied as well.

Even though from manual analysis it could be concluded that the “dummy coreference resolution” which was applied within the sentence extraction system, showed a reasonably good performance, one possible enhancement of the described system could certainly be the use of real coreference resolution. Especially with systems using distant supervision with data different from Wikipedia data, coreference resolution is necessary, for the simple reason that it cannot be assumed that all sentences are likely to refer to the same person, as it is the case in Wikipedia articles which have a person as topic.

Within the setting of a whole relation extraction system, before the classification of the sentences on a previously trained classifier can take place, possible candidate sentences have to be discovered within unstructured text. This can be done by rules, such as if a nationality in connection with a person can be found in a sentence, for the attribute of “nationality”. For find people, named entity recognition is necessary. can be a first hint that a relation might exist. In the following, the candidate sentences which have been extracted can be tested for containing the relation by applying the classifier. As soon as a candidate sentence has been classified for containing a certain relation, the entities which are part of this relation can be extracted and stored in a tuple labelled for this specific relation type.

5 Final Conclusion and Ideas for Future Research

This dissertation was about a system of distant supervision for relation extraction, extracting relations between people and attributes. In the first part, we built a sentence extraction system with the purpose of automatically collecting possible training data for a relation classification system This system has been based on the underlying intuition of distant supervision that whenever a person and an attribute, which are known to stand in a particular relation to each other, appear in one sentence this sentence is likely to express that relation (Mintz et al., 2009). Using the sentence extraction system, data was automatically collected from Wikipedia for a selected sample of four attributes: places lived, place of birth, professions and nationality. Subsequently, through manual analysis strengths and weaknesses of the mechanism in connection with the extraction of people's attributes could be shown. Furthermore, based on findings that for some attributes the extracted data is very noisy, we have made suggestions for improvement by adding additional extraction rules. In the second part of this dissertation, we tested these suggestions within an example system for the attribute of nationality and it could be shown that, indeed, an improvement of the system could be reached.

In short, one of the contributions of the presented work are the tendencies that have been detected within the data obtained through the application of the intuition of the distant supervision approach. These tendencies let to the suggestion of possible improvements of the mechanisms, as the second contribution, adding additional general extraction rules, which, in the conducted experiments, seem to give at least a slight improvement to a system of distant supervision. The chances are high that these, or similar extraction rules can have a positive effect on the general performance of systems using the discussed mechanism, at least in connection with the used data set.

According to the manual analysis of the extracted sentences which are used as training data, the conclusion can be drawn, however, that Wikipedia is at the same time the best and the worst possible resource for training data for this kind of weakly supervised system. In connection with Freebase, it has extreme advantages, which are not shared with any other collection of text as big as Wikipedia. First of all, the data is freely accessible and available in a large quantity, which will still be growing in the future. However, the biggest advantage of Wikipedia in connection with Freebase is that, since the entries in Freebase are structured knowledge from Wikipedia, both resources are very well-adapted to each other which makes it easy to find the retrieved attributes within Wikipedia.

On the other hand, Wikipedia has a lot of disadvantage, which become obvious especially with the attributes focused on. The most important attributes of people are mentioned at the beginning of the article, often in a semi-structured format. This makes the data of Wikipedia very specific with the result that a classifier trained on this data, will have worse results, if used with any other data which is not taken from Wikipedia. Furthermore, as was shown above, it can happen that Infobox data is merged into the beginning of the Wikipedia article, which is hard to remove. This, however, can be a problem of the specific data dump which was used for this project.

One disadvantage of Freebase is that the number of stored relations are limited. Even though many relations are present, it is not an unlimited resource containing all kind of imaginable relation types. However, other databases exist which can also be used for distant supervision and which possibly contain knowledge that cannot be found in Freebase. The Semantic Web[47] is a collaborative project, including several databases, among them Dbpedia[48], which, like Freebase, contains structured information from Wikipedia. The Semantic Web aims at providing data, more specifically, semantic information in the format of relations, in a machine readable uniform format. This machine readable format can be accessed through query languages by computers, which makes it perfect for applications using relations between entities, as for example in order to build a relation extraction system similar to the one described.

In the future, using the mechanism of distant supervision, a system could be imagined which does everything automatically. If a specific system of relation extraction is needed, the system can use web ontologies, for example the ones included in The Semantic Web, in order to retrieve specific relation pairs and build a system for relation extraction for a specific relation type, language independently.

47 http://semanticweb.org/
48 http://dbpedia.org/

Concerning the data for the different attributes which this project focusses on, it can be said that the degree of efficiency of the described mechanism of distant supervision depends to a high degree on the single feature. Whereas for some attributes, as for example "profession" the extracted sentences show an acceptable amount of noise, for others, as for example the attribute of "nationality", the noise is very prominent, to an amount, that no good results can be expected. Even if a very big amount of training data is provided, it can be hard for a system to handle features this noisy. As has been shown, general, unspecific rules which can be added to the underlying intuition of distant supervision, can help to make the data less noisy. The proposed rules were limiting the sentences which are taken from each article to the first three sentences and using different formats encoding "nationality".

In future research, the aim could be to test these rules, especially the first rule of limiting the sentences per article, with other attributes and relation types in order to discover if they can be regarded as generally valid rules. Another aim could be to develop more rules, possibly applicable for a wide range of attributes, which reduce noise in the data. Possible rules, which could be tested in this respect, are restriction of number words between the two entities, restrictions in terms of sentence length and restrictions in terms of sentence complexity.

6 Appendix

6.1 Example Script: Sentence Extraction System

```
#usr/bin/python

import csv
import re
from text_sentence import *
import codecs
import cPickle as pickle

freebase_people = open('/home/tilia/data/person_names.pkl', 'rb')
person_names = pickle.load(freebase_people)
freebase_people.close()
print 'pickle file: list of names was loaded'
# loads the list of peoples' names from freebase

freebase_attributes_list = open('/home/tilia/data/freebase_attributes_list_2.pkl', 'rb')
person_attributes = pickle.load(freebase_attributes_list)
freebase_attributes_list.close()
print 'pickle file: list of attributes was loaded'
# loads the list of peoples' attributes from freebase

ifile = open('/home/data/corpora/wikipedia/wikipedia.en', 'rb')
ifile_reader = csv.reader(ifile, delimiter = '\t')
# a cvs.reader object is returned which serves to read the wikipedia dump
csv.field_size_limit(1000000000)

persistent_countries = open('/home/tilia/data/nationalities.pkl', 'rb')
country_list = pickle.load(persistent_countries)
print 'pickle file: list of countries was loaded'
# loads a list with contries and forms of nationalities


relevant_sentences_nationalities = open('/home/tilia/output_files/rs_nationalities_tagged2.txt',
'wb')
#textfile will contain all relevant sentences

csv_file_nationalities = open('/home/tilia/output_files/relevant_nationalities_10_3lines.csv', 'wb')
relevant_nationalities_writer = csv.writer(csv_file_nationalities, delimiter = '\t')
# csv file will contain the following fields: sentence, relation_tag, person, attribute


def get_relevant_csv (row):
    ''' takes a row of the Wikipedia dump and determines if this row contains an article about a
person or not.
    If it does, it returns the row, if it does not, it returns None'''


    ifile_tester = []
    # empty list to check if a line of the csv document is relevant (= is about a person)
    # this list is necessary because csv has to be read into list in order to be processed

    ifile_tester.append(row)
    ifile_tester_flat = sum(ifile_tester, [])

    # if the title (in the first position of the flattened list) is the name of a person, it can be
assumed that the article is about a person
    if row[0] in person_names:
        print 'function found a person: ', ifile_tester_flat[0]
        return row

    else: return None


def get_article (relevant_csv_row):
    '''takes a row of the csv document which has before been identified as relevant and
    uses the text version of the wikipedia article to split the sentences and clean the data;
    returns a a list which contains the wikipedia article in the format of (cleaned) sentences.'''

    print 'the data of', relevant_csv_row[0], 'is being processed'

    relevant_list = relevant_csv_row[0:2]
    # html version is cut off

    relevant_list_strings = relevant_list[1].rsplit()
    # just the raw text is considered and cut into substrings

    # these regular expressions are defined to delete all'\\n' and all '\t' from the data
    pattern_match_1 = re.compile('(\\\\[n])+')
    pattern_match_2 = re.compile('(\\\\[t])')

    # this new list will serve to store data from which the RE patterns have been removed
    clean_list = []

    # pattern '\\n' is removed from data
    for single_string in relevant_list_strings:
        new_string = pattern_match_1.sub(' ', single_string)
        clean_list.append(new_string)

    # clean_list_1 has all'\\n' and all '\t' removed
    clean_list_1 = []

    # pattern '\t' is removed from data
    for single_string in clean_list:
        new_string = pattern_match_2.sub(' ', single_string)
        clean_list_1.append(new_string)

    # clean_list_1 is joined into a string
    clean_string = ' '.join(clean_list_1)

    # clean_string is being tokenized using the text_sentence module
    t = Tokenizer()
    tokenized_clean_string = list(t.tokenize(clean_string))

    # in list_of_sentences each sublist will be one sentence of the article
    list_of_sentences = []
    # new_list is a temporary list which will contain one sentence and is continuously added to
list_of_sentences
    new_list = []

    for item in tokenized_clean_string:

    # the unicode string is taken from the token object (generated be text_sentence module).
        item_string = item.value

        # the position of the respective token is discovered and assigned to a variable.
        item_position = tokenized_clean_string.index(item)

        # Words at the beginning of the sentences are capitalized unless the token 2 positions before
is an abbreviation.
        if item.is_sent_start:
            if tokenized_clean_string[item_position-2].is_fuzzy_abbr == False and
tokenized_clean_string[item_position-2].is_abbr == False:
                item_string = item_string.capitalize()

        # the string item_string is added to the list which contains the sentence.
        new_list.append(item_string)

        # if the token is marked as sent_end and unless the previous token is marked as abbreviation
new_list is a added to list_of_sentences
        if item.is_sent_end:

            if tokenized_clean_string[item_position-1].is_fuzzy_abbr == False and
tokenized_clean_string[item_position-1].is_abbr == False:
                list_of_sentences.append(new_list)
                # this new list will contain the next sentence
                new_list = []

    # sentences longer than 80 words are excluded, as these normally don't contain useful information
    # sentences containing "excluded_chars" are excluded, as they normally don't contain useful
information
    # every sentence whi
    excluded_chars = ['*','|', u'\u2022']
    excluded_set = set(excluded_chars)

    sentences_counter = 0
    new_sentence_list = []

    for one_sentence_list in list_of_sentences:
        if len(one_sentence_list) > 80:
            pass

        elif excluded_set.intersection(one_sentence_list):
            pass

        elif '\u2022' in one_sentence_list:
            pass

        # sentences which do not contain any of the excluded symbols are added to a new list which is
then returned by the function
        else: new_sentence_list.append(one_sentence_list)

    print sentences_counter, 'sentence counter'
    print len(new_sentence_list), 'is the new length'

    return new_sentence_list


def get_nationality ( name_of_person ):
    '''takes the name of a person and returns all nationalities
```

```
        which can be found for all instances with the same name'''


    nationality_list = []
    #list will serve to store nationalities of all people

    people_counter = 0
    #will serve to count how many people are there with the same name

    for single_person_attributes in person_attributes:
        if name_of_person == single_person_attributes[0]:
            # the name of the person has to be found on the first position of the attributes list

            people_counter = people_counter +1
            # counts how many people there are with the same name

            if not single_person_attributes[2] == '':
                # the nationality is on position 2 in the attributes list

                nationality = single_person_attributes[2]
                nationality_list.append(single_person_attributes[2])
                # all nationalities of all people with the same name are collected in a list

            else:
                pass


    if nationality_list != []:
        nationality_list_new = []
        for item in nationality_list:
            item_new = item.replace(', ', ';'). replace(',', ';').split(';')
            # if there is more than one nationality for a person it has to be split by ','
            print item_new
            for new_string in item_new:
                nationality_list_new.append(new_string)
                # all nationalities for all people are present now


        # IN THE FOLLOWING, DIFFERENT FORMATS OF (THE RELEVANT) NATIONALITIES ARE FOUND

        nationalities = []
        # nationalities will contain lists of all formats of all nationalities

        for nation in nationality_list_new:
            for sub_list in country_list:
                if nation in sub_list[0]:
                    nationalities.append(sub_list)


        relevant_nationalities = sum(nationalities,[])
        # all nationalities are in the same list now

        country_nationality = []
        #list serves to store all "synonyms" of nationality formats

        for one_nationality in relevant_nationalities:
            split_nationality = one_nationality.replace(' or ', ';').replace(', ', ';').split(';')
            #nationality "synonyms" are stored in the list divided by one of the characters above
            country_nationality.extend(split_nationality)


        nationalities_unicode = []
        # list serves to store all nationalities in unicode

        for nationality in country_nationality:
            nationalities_unicode.append(unicode(nationality, 'utf-8'))


        print len(nationality_list), ' nationalities were found for ', people_counter, ' instances
of', name_of_person, ': ', ' '.join(nationality_list_new) , '\n'

        print nationality_list_new, 'nationality list new'
        print nationalities_unicode, 'these nationalities are returned'
        return nationalities_unicode

    elif nationality_list == []:
        print 'no nationality could be found for any instance of ', people_counter, name_of_person
        return None

def get_sentences ( name_of_person, list_of_sentences, one_attribute):
    '''returns relevant sententences (sentences containing the name of a person plus the respective
attribute);
    if no relevant sentences can be found the function returns None'''

    name = []

    relevant_sentences = []
    # this list serves to store all relevant sentences (containing attribute and name) that could be
```

```
found for a person
    relevant_attributes = []
    relevant_name = []

    for item in name_of_person.split():
        name.append(unicode(item, 'utf-8'))

    pronoun_list = [u'He', u'he',u'him', u'She', u'she', u'her']
    # pronoun list for "dummy coreference resolution"

    pronoun_list.extend(name)
    #name and pronoun are now in the same list

    pronoun_set = set(pronoun_list)
    attribute_set = set(one_attribute)

    tagged_relevant_sentences = []

    for one_sentence_list in list_of_sentences:

        if pronoun_set.intersection(one_sentence_list):

            if attribute_set.intersection(one_sentence_list):
                relevant_attributes.extend([x for x in
attribute_set.intersection(one_sentence_list)])
                #forms of relevant attributes (which were found in relevant sentences) are extracted
into a list


                relevant_sentences.append(one_sentence_list)
                relevant_name.extend([x for x in pronoun_set.intersection(one_sentence_list)])
                # the exact occurence of the name or pronoun in the sentence is discovered


        else: pass
        #if the name cannot be found in a sentence, nothing happens


    print len(relevant_sentences), 'sentences could be found for', name_of_person


    return (relevant_sentences, relevant_attributes, relevant_name)
    #the relevant sentences are returned, together with attribute and name(or pronoun) as they could
be found in the sentence


line_counter = 0
# optional counter counts how many lines of the csv file are considered
people_counter = 0
# counts how many people are considered
no_sentences = 0
#counts for how many people no sentences could be found (even though a nationality could be found)
no_nationality = 0
#counts for how many people no nationality could be found

relevant_sentence_counter = 0

for i, row in enumerate(ifile_reader):
    if relevant_sentence_counter < 5000:
    #5000 sentences will be extracted

        print i
        # keeps track of how many lines of the Wikipedia dump have been considered

        relevant_row = get_relevant_csv(row)
        # uses function (defined above) to get a relevant Wikipedia article

        if relevant_row != None:
        #when a relevant row is found, the attribute and possible sentences can be discovered
            line_counter = line_counter + 1
            people_counter = people_counter + 1


            name_of_person = row[0]


            list_of_sentences = get_article(row)
            # uses function (defined above) to tokenize and preprocess the article

            nationality = get_nationality(name_of_person)
            # uses function (defined above) to get the nationality from Freebase

            if nationality != None:

                relevant_tuple = get_sentences(name_of_person, list_of_sentences, nationality)
                relevant_sentences = relevant_tuple[0]
                relevant_attributes = relevant_tuple[1]
                relevant_names = relevant_tuple[2]
                # the function (defined above) is used to extract a list of all relevant sentences
for a person


                if relevant_sentences != []:
                    relevant_sentences_counter = 0

                    for one_relevant_sentence in relevant_sentences:

                        u_relevant_sentence = []
                        relevant_sentences_counter = relevant_sentences_counter +1
                        # counts how many relevant sentences could be found

                        for one_word in one_relevant_sentence:
                            u_relevant_sentence.append(one_word.encode('utf-8'))

                        u_relevant_name = []
                        for name_part in relevant_names:
                            u_relevant_name.append(name_part.encode('utf-8'))
                            # (unicode handling)



                        print ' '.join(u_relevant_sentence)
                        print '\n'
                        relevant_sentence_counter = relevant_sentence_counter + 1
                        relevant_sentences_nationalities.write(' '.join(u_relevant_sentence) +
'<relation_nat>' +'\n')
                        relevant_nationalities_writer.writerow([' '.join(u_relevant_sentence)]  +
['class_nat'] + [name_of_person] + [''.join(u_relevant_name)] + [relevant_attribute.encode('utf-8')])
                        #relevant sentence, class label, name of the person, name as it appears in
sentence, attribute as it appears in sentence
                        # everything is written into the csv-file



                elif relevant_sentences == []:
                    print 'the list of relevant sentences for this person is empty for this
attribute'
                    no_sentences = no_sentences + 1
                    print 'so far no sentences could be found for number of people: ', no_sentences
                    print '\n'

            elif nationality == None:
                print 'no nationality'
                no_nationality = no_nationality + 1
                print 'so far no nationality was found for number of people ', no_nationality
                print '\n'

        else:
            line_counter = line_counter + 1


    else: break
print 'several formats'


print relevant_sentence_counter, ' relevant sentences were found for ', people_counter, ' people'
relevant_sentences_nationalities.close()
```

6.2 Script: Machine Learning System

```
#!/usr/bin/python
# coding: utf-8


import csv
from text_sentence import *
import codecs
import cPickle as pickle
import nltk
import random
from nltk.classify import apply_features
from nltk.classify import NaiveBayesClassifier
import nltk.metrics



csv_file_positive = open('/Users/tilia/desktop/ml_data/1_run_countries/pos_country_names.csv', 'rb')
positive_reader= csv.reader(csv_file_positive, delimiter = '\t')
```

```
# a cvs.reader object is returned which serves to read a csv file containing the relevant sentences
# format: relevant_sentence class_nat name_of_person name_in_sentence nationality_in_sentence
csv.field_size_limit(1000000000)



csv_file_negative = open('/Users/tilia/desktop/ml_data/1_run_countries/neg_nat_countries.csv', 'rb')
negative_reader = csv.reader(csv_file_negative, delimiter = '\t')
csv.field_size_limit(1000000000)
# format of the negative file:
#relevant_sentence nat_negative name_of_person name in sentence (if more than one, divided by comma)
countries in the sentence (if more than one, divided by comma)


csv_file_testing = open('/Users/tilia/desktop/ml_data/all_testing.csv', 'rb')
testing_reader = csv.reader(csv_file_testing, delimiter = '\t')
csv.field_size_limit(1000000000)


def csv_to_tuple(csv_reader):
    '''takes a csv - file and returns a list of tuples of the format (relevant_sentence,
relation_type_label, name_in_sentence, attribute_in_sentence)'''
    tuple_list = []
    try:
        for i, row in enumerate(csv_reader):
            a_tuple = (row[0], row[1], row[3], row[4])
            tuple_list.append(a_tuple)


    except IndexError: print 'index error'

    return tuple_list


def tokenize_sentence(sentence):
    '''Tokenizes a sentence using the text-sentence tokenizer.'''
    t = Tokenizer()

    list_of_sentences = []
    new_list = []


    tokenized_string = list(t.tokenize(sentence))

    for item in tokenized_string:
    # the unicode string is taken from the token object and transformed to unicode.
        item_string = unicode(item.value)
        item_string = item.value
        # the position of the respective token is discovered and assigned to a variable.
        item_position = tokenized_string.index(item)
        # Words at the beginning of the sentences are capitalized unless the token 2 positions before
is an abbreviation.
        if item.is_sent_start:
            try:
                if tokenized_string[item_position-2].is_fuzzy_abbr == False and
tokenized_string[item_position-2].is_abbr == False:
                    item_string = item_string.capitalize()
            except IndexError: pass
        # the string item_string is added to the list which contains the sentence.
        new_list.append(item_string)
        # if the token is marked as sent_end and unless the previous token is marked as abbreviation
new_list is a added to list_of_sentences
        if item.is_sent_end:
            if tokenized_string[item_position-1].is_fuzzy_abbr == False and
tokenized_string[item_position-1].is_abbr == False:
                list_of_sentences.append(new_list)
                # this new list will contain the next sentence
                new_list = []

    return(list_of_sentences)



def entities_to_tags(sentence_tokenized, name_list, attribute):
    ''' transformes name and attribute in the sentence for the tags '<name>' and '<attr>;
    returns the sentence as a list of words with name and attribute transformed'''

# first the name of the person (with up to 5 words) is transformed
# then the attribute (with up to 4 words) is transformed

    sentence_copy = sentence_tokenized[:]
    label_list = ['<name>', '<attribute>']
    label_counter = 0
    pronoun_list = ['he','He', 'She', 'she', 'his', 'her']

    relevant_name_list = name_list.split(',')
    #print relevant_name_list, 'relvantname'
    relevant_name_unicode = []
```

```
for name_part in relevant_name_list:
    unicode_name = unicode(name_part, 'utf-8')
    relevant_name_unicode.append(unicode_name)

for position, word in enumerate(sentence_tokenized):
    if word in relevant_name_unicode:
        sentence_copy.pop(position)

        next_word_position = position + 1
        if sentence_tokenized[next_word_position] in relevant_name_unicode:
            sentence_copy.pop(position)

            next_word_position = next_word_position +1
            if sentence_tokenized[next_word_position] in relevant_name_unicode:
                sentence_copy.pop(position)

                next_word_position = next_word_position +1
                if sentence_tokenized[next_word_position] in relevant_name_unicode:
                    sentence_copy.pop(position)

                    next_word_position = next_word_position +1
                    if sentence_tokenized[next_word_position] in relevant_name_unicode:
                        sentence_copy.pop(position)

                    else: sentence_copy.insert(position, '<name>')
                    break

                else: sentence_copy.insert(position, '<name>')
                break

            else: sentence_copy.insert(position, '<name>')
            break

        else:
            sentence_copy.insert(position, '<name>')
            break

    else: pass

if not '<name>' in sentence_copy:

    for pronoun in pronoun_list:
        if pronoun in sentence_copy:
            position = sentence_copy.index(pronoun)
            sentence_copy.pop(position)
            sentence_copy.insert(position, '<name>')
            break
        else: pass



relevant_attribute_list = attribute.split(',')
relevant_attribute_unicode = []
for attribute_part in relevant_attribute_list:
    unicode_attribute = unicode(attribute_part, 'utf-8')
    relevant_attribute_unicode.append(unicode_attribute)


new_sentence = sentence_copy[:]
# sentence copy makes sure that list is not changed while it is used

for position, word in enumerate(new_sentence):
    if word in relevant_attribute_unicode:
        sentence_copy.pop(position)

        next_word_position = position + 1
        if sentence_tokenized[next_word_position] in relevant_attribute_unicode:
            sentence_copy.pop(position)

            next_word_position = next_word_position +1
            if sentence_tokenized[next_word_position] in relevant_attribute_unicode:
                sentence_copy.pop(position)

                next_word_position = next_word_position +1
                if sentence_tokenized[next_word_position] in relevant_attribute_unicode:
                    sentence_copy.pop(position)

                else: sentence_copy.insert(position, '<attribute>')
                break


            else: sentence_copy.insert(position, '<attribute>')
            break

        else:
            sentence_copy.insert(position, '<attribute>')
            break
        else: pass


    return sentence_copy


def preprocess_tuple(raw_tuple):
    '''takes a tuple of the format (relevant_sentence, relation_type_tag, name_in_sentence,
attribute_in_sentence)
    and returns a tuple of the format (relevant_sentence_tokenized, relevant_sentence_pos_tagged,
relation_tag, name, attribute, sentence_with_name_and_attribute_exchanged)'''

    a_sentence, relation_tag, name, attribute = one_tuple
    #tuple is unpacked
    tokenized_sentence = tokenize_sentence(a_sentence)
    #sentence is tokenized using the tokenize_sentence function defined above

    tokenized_sentence_flat = sum(tokenized_sentence, [])

    pos_tagged = nltk.pos_tag(tokenized_sentence_flat)
    # part of speech tags are added to the tokenized sentence

    tagged_entities = entities_to_tags(tokenized_sentence_flat, name, attribute)
    #the entities within the sentences are transformed to labels (by the function defined above)

    new_tuple = tokenized_sentence_flat, pos_tagged, relation_tag, name, attribute, tagged_entities
    # new (preprocessed) tuple is created and returned

    return new_tuple


def return_indexes(list_a, list_b):
    '''finds one list in another list and returns indexes as a tuple'''
    for i in xrange(len(list_a)):
        if list_a[i:i+len(list_b)] == list_b:
            indexes = (i,i+len(list_b))
    return indexes


def remove_punctuation(a_list):
    tags = ['<attribute>', '<name>']
    punctuation_articles = [',', '+', '.', '-', ' ', '(', ')', 'the', 'a', '!']
    removable_items = punctuation_articles + tags
    new_list = []
    for word in a_list:
        if not word in removable_items:
            new_list.append(word)
    return new_list

def bag_of_words_features(one_sentence_list):

    return {'bag_of_words': tuple(one_sentence_list)}



def extract_features(preprocessed_tuple, list_all_words):
    '''extracts different features from the sentences'''

    tagged_entities = preprocessed_tuple[5]
    pos_tagged = preprocessed_tuple[1]

    features = {}
    entity_tags = ['<name>', '<attribute>']

    words_between_entities = []

    tuple_with_tags = preprocessed_tuple[5]
    sentence_words = set(tuple_with_tags)
    features = {}
    bigrams_whole_sentence = nltk.NgramModel(2,tuple_with_tags)
    bigrams_sentence_set = bigrams_whole_sentence._ngrams

    for word in list_all_words:
        features['contains(%s)' % word] = (word in sentence_words)
        #builds dictionary with the name of the feature ('contains word ...') as key and the actual
word as value
    for bigram in list_all_bigrams:
        features['contains(%s)'%str(bigram)] = (bigram in bigrams_sentence_set)



    try:
        first_entity_index = tagged_entities.index('<name>')
        second_entity_index = tagged_entities.index('<attribute>')

        if first_entity_index < second_entity_index:
            words_between = tagged_entities[first_entity_index+1:second_entity_index]
            #print words_between, 'words between'
        if first_entity_index > second_entity_index:
            words_between = tagged_entities[second_entity_index+1:first_entity_index]
            #print words_between, 'words between'

        words_between_set = set(words_between)
        bigrams_between_words = nltk.NgramModel(2,words_between)
        bigrams_between_set = bigrams_between_words._ngrams

        pos_between = []

        len_words_between = len(words_between)

        word_list = []
        common_pos = []
        pos_list = []

        for word, tag in pos_tagged:
            word_list.append(word)
            pos_list.append(tag)


        sequence_indexes = return_indexes(word_list, words_between)


        i_first,i_second = sequence_indexes

        pos_tagged_bw = pos_tagged[i_first:i_second]

        pos_tagged_after = pos_tagged[i_second:i_second+4]

        pos_tagged_before = pos_tagged[i_first-4:i_first]


        pos_tag_after = []
        pos_tag_bw = []
        pos_tag_before = []
        for word, pos_tag in pos_tagged_after:
            pos_tag_after.append(pos_tag)
        pos_tag_bw = []
        for word, pos_tag in pos_tagged_bw:
            pos_tag_bw.append(pos_tag)
        pos_tag_before = []
        for word,pos_tag in pos_tagged_before:
            pos_tag_before.append(pos_tag)

        pos_all_between = nltk.NgramModel(len(words_between), words_between)
        pos_all_between_set = pos_all_between._ngrams

        pos_bigrams_before = nltk.NgramModel(2, pos_tag_before)
        pos_bigrams_before_set = pos_bigrams_before._ngrams

        pos_bigrams_after = nltk.NgramModel(2,pos_tag_after)
        pos_bigrams_after_set = pos_bigrams_after._ngrams

        pos_bigrams = nltk.NgramModel(2,pos_tag_bw)
        bigrams_pos_set = pos_bigrams._ngrams

        pos_tag_between_set = set(pos_tag_bw)

        for pos_tag in list_all_pos_bigrams:
            features['pos bigram before first entity(%s)' % str(pos_tag)] = (pos_tag in
pos_bigrams_before_set)
            #part of speech bigram features for 4 words before the first entity

        for pos_tag in list_all_pos_bigrams:
            features['pos bigram after second entity(%s)' % str(pos_tag)] = (pos_tag in
pos_bigrams_after_set)
            # part of speech bigram features for 4 words after the second entity

        for pos_tag in list_all_pos_bigrams:
            features['pos bigram between entities(%s)' % str(pos_tag)] = (pos_tag in bigrams_pos_set)
            # part of speech bigram features for sequence of words between the two entities

        for word in list_all_words:
            features['word between entities(%s)' % word] = (word in words_between_set)
            # bag of word features for words between entities

        for bigram in list_all_bigrams:
            features['bigram between entities(%s)' % str(bigram)] = (bigram in bigrams_between_set)
            # bigram features for words between entities


    except ValueError:
        print 'wrongly tagged entities', tagged_entities
        # wrong tags can be assigned in exceptional cases, when the name and the country are the same
word in negative sentences.

        wrongly_tagged = tagged_entities
        substitude_words = []

        for word in wrongly_tagged:
            if word == '<name>':
                pass
            if word == '<attribute>':
                pass
            else: substitude_words.append(word)

        #length_sentence = len(tagged_entities)
        substitude_set = set(substitude_words)
        for word in list_all_words:
            features['word between entities(%s)' % word] = (word in substitude_set)
    #
    return features


positive_tuple_list_full = csv_to_tuple(positive_reader)
# the csv file is transformed into a list of tuples
random.shuffle(positive_tuple_list_full)
positive_tuple_list = positive_tuple_list_full[0:250]

print 'from', len(positive_tuple_list_full), ', considered positive tuples:',
len(positive_tuple_list)

tuple_counter = 0
# counts the number of tuples

positive_collection = []
#list will store all tuples from the positive data set
positive_collection_pos = []
positive_preprocessed_tuples = []

for one_tuple in positive_tuple_list:
    tuple_counter = tuple_counter + 1

    preprocessed_tuple = preprocess_tuple(one_tuple)
    labeled_preprocessed_tuple = preprocessed_tuple, one_tuple[1]
    positive_preprocessed_tuples.append(labeled_preprocessed_tuple)

    tagged_sentence = preprocessed_tuple[5], one_tuple[1]
    # gives a tuple with the (tokenized and entity-tagged) sentence and the label that it is positive
    positive_collection.append(tagged_sentence)

    pos_sentence = (preprocessed_tuple[1], preprocessed_tuple[5],one_tuple[1])
    positive_collection_pos.append(pos_sentence)


negative_tuple_list_full = csv_to_tuple(negative_reader)
#the negative csv files is transformed into a list of tuples

random.shuffle(negative_tuple_list_full)
negative_tuple_list = negative_tuple_list_full[0:250]
print 'from', len(negative_tuple_list_full), ', considered negative tuples:',
len(negative_tuple_list)

negative_collection = []
negative_collection_pos = []
negative_preprocessed_tuples = []

for one_tuple in negative_tuple_list:
    tuple_counter = tuple_counter + 1
    #print one_tuple
    preprocessed_tuple = preprocess_tuple(one_tuple)
    #returns a tuple of the format
    #(relevant_sentence_tokenized, relevant_sentence_pos_tagged, relation_tag, name, attribute,
sentence_with_name_and_attribute_exchanged)
    labeled_preprocessed_tuple = preprocessed_tuple, one_tuple[1]
    negative_preprocessed_tuples.append(labeled_preprocessed_tuple)

    tagged_sentence = preprocessed_tuple[5], one_tuple[1]
    #gives a tuple with the (tokenized and entity-tagged) sentence and the tag of the relations
    negative_collection.append(tagged_sentence)
    pos_sentence = (preprocessed_tuple[1], preprocessed_tuple[5],one_tuple[1])
    negative_collection_pos.append(pos_sentence)

testing_tuple_list_full = csv_to_tuple(testing_reader)
random.shuffle(testing_tuple_list_full)
testing_tuple_list = testing_tuple_list_full[0:300]
print 'from', len(testing_tuple_list_full), ', considered testing tuples:', len(testing_tuple_list)

testing_collection = []
testing_collection_pos = []
testing_preprocessed_tuples = []

for one_tuple in testing_tuple_list:
    tuple_counter = tuple_counter + 1
    preprocessed_tuple = preprocess_tuple(one_tuple)
    #returns a tuple of the format
    #(relevant_sentence_tokenized, relevant_sentence_pos_tagged, relation_tag, name, attribute,
sentence_with_name_and_attribute_exchanged)
    labeled_preprocessed_tuple = preprocessed_tuple, one_tuple[1]
    testing_preprocessed_tuples.append(labeled_preprocessed_tuple)

    tagged_sentence = preprocessed_tuple[5], one_tuple[1]
    #gives a tuple with the (tokenized and entity-tagged) sentence and the tag of the relations
    testing_collection.append(tagged_sentence)
    pos_sentence = (preprocessed_tuple[1], preprocessed_tuple[5],one_tuple[1])
    testing_collection_pos.append(pos_sentence)


whole_collection = positive_collection + negative_collection
random.shuffle(whole_collection)
all_words_list = []
all_pos_list= []
for a_tuple in whole_collection:
    all_words_list.extend(a_tuple[0])
# makes a list of all the words of all sentences (positive and negative)

all_words_new = remove_punctuation(all_words_list)
#punctuation, articles and entity labels are removed
all_words = nltk.FreqDist(w.lower() for w in all_words_new)
#measures the frequency of every word that occurs in both sets
frequent_words = all_words.keys()[:1000]
# the 1000 most frequent words are considered
print len(frequent_words), 'words are considered'


bigram_model = nltk.NgramModel(2,all_words_new)
list_all_bigrams = list(bigram_model._ngrams)
# a bigram model is built for all sentences


whole_collection_pos = positive_collection_pos + negative_collection_pos
random.shuffle(whole_collection_pos)
all_pos_tags = []
#contains all pos_tags
for a_tuple in whole_collection_pos:
    for (word, tag) in a_tuple[0]:
        all_pos_tags.append(tag)

pos_tag_list = []
#contains every pos_tag just once
irrelevant_pos = ['.', ':', ',', '-NONE-','#','``']
for pos_tag in all_pos_tags:
    if not pos_tag in pos_tag_list:
        if not pos_tag in irrelevant_pos:
            pos_tag_list.append(pos_tag)


all_pos_bigrams = nltk.NgramModel(2,all_pos_tags)
#contains bigrams for all part of speech tags
list_all_pos_bigrams = list(all_pos_bigrams._ngrams)

print 'feature extraction starts now'

positive_features = [(extract_features(n, frequent_words),g) for (n,g) in
positive_preprocessed_tuples]
negative_features = [(extract_features(n, frequent_words),g) for (n,g) in
negative_preprocessed_tuples]
testing_features = [(extract_features(n, frequent_words),g) for (n,g) in testing_preprocessed_tuples]


training_set = negative_features + positive_features
testing_set = testing_features
#data is divided into training set and testing set

random.shuffle(training_set)
random.shuffle(testing_set)

print 'train on %d instances, test on %d instances' % (len(training_set), len(testing_set))[49]


classifier = nltk.NaiveBayesClassifier.train(training_set)
# classifier is trained on training set


print 'accuracy: ', nltk.classify.accuracy(classifier, testing_set)


classifier.show_most_informative_features(20)
```

49 Some ideas are taken from: http://streamhacker.com/2010/05/24/text-classification-sentiment-analysis-stopwords-collocations/

7 Bibliography

7.1 References

Adrian, B., Neumann, G., Troussov, A. and Popov, B. (Eds.). 2008. *Proceedings of the KI 2008 Workshop on Ontology- Based Information Extraction Systems.* Available from: <http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-400/> 5 May 2011.

Agichtein, E. and Gravano, L.. 2000. *Snowball: Extracting Relations from large Plain-Text Collections.* In Proceedings of the 5th ACM International Conference on Digital Libraries, San Antonio, TX (2000), pp. 85–94

Bach, N. and Badaskar, S.. 2007. *A Review of Relation Extraction*. Technical report, Language Technologies Institute, Carnegie Mellon University.

Banko, M., Etzioni, O., Soderland, S. and Weld, D.S.. 2007. *Open information extraction from the web*. In: Commun. ACM 51, 12(December 2008), pp. 68-74.

Banko, M. and Etzioni, O.. 2008. *The Tradeoffs Between Open and Traditional Relation Extraction*. In: Proceedings of ACL-08: HLT (June 2008), pp. 28-36.

Bird, S., Klein, E., Loper, E.. 2009. *Natural Language Processing with Python*. CA: Oreilly & Associates Inc..

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R. and Hellmann, S.. 2009. *DBpedia - a Crystallization Point for the Web of Data.* Web Semantics: Science, Services and Agents on the World Wide Web, pp. 154–165.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T. and Taylor, J.. 2008. *Freebase: a collaboratively created graph database for structuring human knowledge.* In: SIGMOD ’08. New York: NY. ACM, pp. 1247-1250.

Brin, S.. 1998. *Extracting Patterns and Relations from the World Wide web.* In: Proceedings of WebDB Workshop at 6th International Conference on Extending Database Technology (WebDB'98).

Bunescu, R. and Mooney, R. J.. 2005. *Subsequence kernels for relation extraction. Neural Information Processing Systems.* In Advances in Neural Information Processing Systems 18 (2006), pp. 171-178.

Bunescu, R. and Mooney, R. J.. 2007. *Learning to Extract Relations from the Web using Minimal Supervision.* Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 576–83.

Culotta, A. and Sorensen, J.. 2004. *Dependency tree kernels for relation extraction.* In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics ACL 04, pp. 423–429.

Google Insights. Available from: <http://www.google.com/insights/search/> 7 May 2012.

Grishman, R. and Sundheim, B. 1996. *Message understanding conference* - 6: A brief history. In: Proceedings of the 16th International Conference on Computational Linguistics (COLING), pages 466-471.

Grishman, R.. 2003. *Information Extraction*. In: Mitkov, R. (Ed.). 2003. The Oxford Handbook of Computational Linguistics. Oxford: Oxford University Press, pp. 553-560.

Grishman, R.. 2010. I*nformation extraction.* In: Clark, A.; Fox, C. & Lappin, S. (Eds.) The Handbook of Computational Linguistics and Natural Language Processing. London: Wiley- Blackwell, pp. 517 – 530.

Han, X. and Zhao, J.. CASIANED: *People Attribute Extraction based on Information Extraction.* In: 2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference.

Hasegawa, T., Sekine, S. and Grishman, R.. 2004. *Discovering relations among named entities from large corpora.* In: Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume, pp. 415–22.

Hearst, M.. 1992. *Automatic acquisition of hyponyms from large text corpora.* In: Fourteenth International Conference on Computational Linguistics, Nantes, France, pp. 1083–1106.

Jurafsky, D. and Martin, H. M.. 2007. *Information Extraction. In: Speech and Language Processing – An introduction to natural language processing, computational linguistics, and speech recognition.* London: Pearson Education Ltd.. pp. 831-877.

Kambhatla, N.. 2004. *Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations.* Proceedings of the ACL 2004.

Labsky, M., Svatek, V., Nekvasil, M.. 2008. *Information Extraction Based on Extraction Ontologies: Design, Deployment and Evaluation.* In: Proc. of the KI 2008 Workshop on Ontology- Based Information Extraction Systems.

Lan, M., Zhang, Y.Z., Lu, Y., Su, J. and Tan, C.L.. 2009. *Which who are they? People attribute extraction and disambiguation in web search results.* In: 2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference.

Medelyan, O., Milne, D,, Legg, C. et al.. 2009. *Mining Meaning from Wikipedia.* In: International Journal of Human-Computer Studies, Volume 67, pp. 716-754

Mintz, M., Bills, S., Snow,R. and Jurafsky, D.. 2010. *Distant supervision for relation extraction without labeled data.* In: ACL '10 Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 1003-1011.

Mitkov, R.. 2010. *Discourse Processing.* In: Clark, A.; Fox, C. & Lappin, S. (Eds.) The Handbook of Computational Linguistics and Natural Language Processing. London: Wiley- Blackwell, pp. 599 – 630.

Moens, M.-F.. 2006. *Information extraction: algorithms and prospects in a retrieval context.* Dordrecht: Springer.

Nguyen, D. P., Matsuo, Y., and Ishizuka, M.. 2007a. *Subtree mining for relation extraction from Wikipedia.* In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers. Rochester, New York: Association for Computational Linguistics, pp. 125– 128.

Nguyen, D. P. , Matsuo, Y. and Ishizuka, M.. 2007b. *Exploiting syntactic and semantic information for relation extraction from Wikipedia.* In: IJCAI07-TextLinkWS, pp. 687–701.

Sarawagi, S.. 2007. *Information Extraction.* In: Foundations and Trends in Databases. Vol. 1, No. 3 (2007), pp. 261-377.

Snow, R., Jurafsky, D. and Ng, A.Y.. 2005. *Learning syntactic patterns for automatic hypernym discovery.* In: Lawrence K. Saul, Yair Weiss, and Le ́on Bottou (eds), NIPS 17, MIT Press, pp. 1297–1304.

Wang, G., Yu, Y. and Zhu, H.. 2007. *PORE: Positive-only relation extraction from Wikipedia text.* In: Proceedings of the Sixth International Semantic Web Conference and Second Asian Semantic Web Conference, ISWC/ASWC'07, Busan, South Korea, pp. 580–594.

WePS Webpage. Available from: <http://nlp.uned.es/weps/index.php/> 4 May 2012.

Wu, F. and Weld, D.. 2007. *Autonomously semantifying Wikipedia.* In: Proceedings of the 16th ACM Conference on Information and Knowledge Management, pp. 41–50.

Wu, F. and Weld, D. S.. 2010. *Open information extraction using Wikipedia.* In: ACL '10 Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 118-127.

Yarowsky, D.. 1995. *Unsupervised Word Sense Disambiguation Rivalling Supervised Methods.* In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, pp. 189 -196.

Zelenko, D., Aone, C., and Richardella, A.. 2003. *Kernel methods for relation extraction.* Journal of Machine Learning Research 3 (2003), pp. 1083–1106

Zhao, S. and Grishman, R.. 2005. *Extracting relations with integrated information using kernel methods.* Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 419–426.

Zhang, M., Su, J., Wang D, Guodong, Z., and Chew Lim Tan. 2005. *Discovering Relations Between Named Entities from a Large Raw Corpus Using Tree Similarity-based Clustering.* In: Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP-05), Jeju Island, Korea, pp. 378–389

Zhou, G., Su, J., Zhang, J. and Zhang, M.. 2005. *Exploring various knowledge in relation extraction.* In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 419–444.

Zhou, G., Zhang, M., Ji, D. and Zhu, Q.. 2007. *Tree kernel-based relation extraction with context- sensitive structured parse tree information.* In: EMNLP/CoNLL 2007, pp. 728–739.

7.2 Resources

Freebase. Available from: <http://www.freebase.com/> 6 May 2011.

Lujo, R.. 2010. *Text-sentence 0.14*. Available from: <http://pypi.python.org/pypi/text-sentence/> 16 October 2010.

Python 2.7.1. . Available from: <http://www.python.org/> 19 November 2011.

Wikipedia. Available from: <www.wikipedia.org> 5 March 2011.

NLTK – Natural Language Toolkit for Python. Available from <**http://www.nltk.org/**> 05 April 2012.