FactRunner: Fact Extraction over Wikipedia

Rhio Sutoyo[1] and Christoph Quix[2,3] and Fisnik Kastrati[2]
[1]*King Mongkut's University of Technology North Bangkok, Thai-German Graduate School of Engineering, Bangkok, Thailand*
[2]*RWTH Aachen University, Information Systems and Databases, Aachen, Germany*
[3]*Fraunhofer Institute for Applied Information Technology FIT, St. Augustin, Germany*

Keywords: Information Extraction, Semantic Search.

Abstract: The increasing role of Wikipedia as a source of human-readable knowledge is evident as it contains an enormous amount of high quality information written in natural language by human authors. However, querying this information using traditional keyword based approaches requires often a time-consuming, iterative process to explore the document collection to find the information of interest. Therefore, a structured representation of information and queries would be helpful to be able to directly query for the relevant information. An important challenge in this context is the extraction of structured information from unstructured knowledge bases which is addressed by Information Extraction (IE) systems. However, these systems struggle with the complexity of natural language and produce frequently unsatisfying results.
In addition to the plain natural language text, Wikipedia contains links between documents which directly link a term of one document to another document. In our approach for fact extraction from Wikipedia, we consider these links as an important indicator for the relevance of the linked information. Thus, our proposed system *FactRunner* focusses on extracting structured information from sentences containing such links. We show that a natural language parser combined with Wikipedia markup can be exploited for extracting facts in form of triple statements with a high accuracy.

1 INTRODUCTION

For the past three decades, relational database systems (RDBMS) have gained great popularity for data management in various application areas. Nevertheless, with the introduction of the World Wide Web (WWW), a huge amount of data repositories in form of unstructured and semi-structured sources became available (e.g., webpages, newspaper articles, blogs, scientific publication repositories, etc). Such sources often contain useful information and are designed to be read by humans, and not by machines. Wikipedia is a good example of this case; with more than 3.8 million articles, it has become one of the main sources of human-readable knowledge repository in the modern information era.

Keyword search is still the dominating way of searching in such large document collections. Although keyword search is quite efficient from a system-oriented point-of-view (relevant documents can be found very quickly using a traditional keyword-based search engine), finding a certain piece of information requires often an iterative, time-consuming process of keyword search. An initial set of keywords is tried first; if relevant documents are returned, then the user has to read parts of the retrieved documents in order to find the information of interest. If the answer is not found, the user has to refine her query, and look for other documents which might contain the answer. This process of querying, reading, and refining might have to be repeated several times until a satisfying answer is found.

Therefore, the search process would be simplified if it could be supported, at least partially, by some system that actually understands the semantics of the searched data. Such systems are called semantic search engines and various approaches have been proposed (Mangold, 2007; Dong et al., 2008). They range from approaches which replace the original keywords entered by the user with semantically related terms (e.g., (Burton-Jones et al., 2003)) to more complex approaches which require a query in a formal language as well as semantically annotated data (e.g., SPARQL endpoints for Linked Data (Heath and Bizer, 2011)).

The latter search paradigm is geared towards data

Sutoyo R., Quix C. and Kastrati F..
FactRunner: Fact Extraction over Wikipedia.
DOI: 10.5220/0004375604230432
In *Proceedings of the 9th International Conference on Web Information Systems and Technologies* (WEBIST-2013), pages 423-432
ISBN: 978-989-8565-54-9

retrieval as the system is only able to answer concepts or documents which contain semantic annotations. This is fine as long as the documents have been annotated with semantic information (e.g., RDF statements). However, most information which is available today is stored in unstructured text documents which do not contain semantic annotations. Web documents, in contrast to plain text documents, contain other useful information such as links, tables, and images. Especially, links connecting documents can be exploited to extract more accurate information from text documents. We apply this idea to the Wikipedia collection.

The task of converting information contained in document collections into formal knowledge is addressed in the research area of Information Extraction (IE), but it is still an unsolved problem. Approaches such as Open Information Extraction (Banko et al., 2007) are able to extract information in form of triples (e.g., statements of the form subject – predicate – object) from unstructured documents, but the extracted triples require more consolidation, normalization and linkage to existing knowledge to become useful for queries. In order to fulfill such an aim at a large scale, this task has to be done in unsupervised and fully-automatic manner. A particular challenge for information extraction from Wikipedia articles is the lack of redundancy of sentences describing a particular fact. Redundancy has been greatly leveraged by systems that perform IE over the web (e.g., TextRunner (Banko et al., 2007) and KnowItAll (Etzioni et al., 2004)) to increase their quality of extracted facts. Furthermore, redundancy is important for the scoring model as frequently occurring facts get a higher score, i.e., these facts are assumed to be correctly extracted with a high probability. A strong scoring model is fundamental towards ensuring the accuracy of the extractor (i.e., for filtering out incorrect triples). This is not the case with Wikipedia, as articles generally address only one topic, and there are no other articles addressing the same topic. This feature reduces information redundancy greatly; if a fact is missed while extracting information from one article, the probability is very low that the same fact will be encountered in other articles.

The contribution of this paper is a novel system called *FactRunner* for open fact extraction from Wikipedia. The primary goal of this system is to extract high quality information present in Wikipedia's natural language text. Our approach is complementary to other approaches which extract structured information from Wikipedia infoboxes (Weld et al., 2009) or the category system (Hoffart et al., 2011). Our method utilizes the existing metadata present in Wikipedia articles (i.e., links between articles) to extract facts with high accuracy from Wikipedia's English natural language text. The metadata is also used in our scoring model and provides higher scores to facts which are based on metadata. Extracted facts are stored in form of triples `(subject, predicate, object)`, where a triple is a relation between subjects and objects that is connected by predicates (Rusu et al., 2007). These triples can then later be queried using structured queries, and thereby enabling the integration of structured and unstructured data (Halevy et al., 2003).

The paper is structured as follows. We will first present in section 2 the main components of our system. Section 3 describes the preprocessing step of our system, which actually does most of the work for the triple extraction, as it simplifies and normalizes the natural language text input. Because of this simplification, the triple extraction method (section 4) is quite simple compared to the preprocessing steps. Section 5 then presents the evaluation results for our system which we applied to a sample of documents from Wikipedia. Related work is discussed in section 6 before we conclude our paper.

2 SYSTEM OVERVIEW

Wikipedia contains a vast amount of information stored in natural text. There is much valuable information hidden in such text, but computers cannot directly reason over the natural text. Wikipedia is designed to be read by humans. To tackle this problem, we introduce a solution which utilizes Wikipedia metadata to detect high quality sentences in the Wikipedia data and extract valuable facts from those sentences. Metadata surrounds important text fragments located in Wikipedia articles. Metadata provides a link pointing to separate Wikipedia articles devoted to the highlighted text, this way giving more information (e.g., general definition, biography, history, etc.), and emphasizing the importance of such text. Wikipedia contributors have invested efforts to create such metadata, and they inserted them manually, giving a good hint on importance of such fragments. To this end, we treat metadata as an indicator of importance of sentences where they occur, and we invest efforts into exploiting such sentences, with the ultimate goal of producing high quality triples.

The system architecture of our system is represented in Figure 1. The architecture consists of six components, namely, the data source, the preprocessor, the extractor, the triple finalizer, the scoring, and the result producer. The main components are the pre-

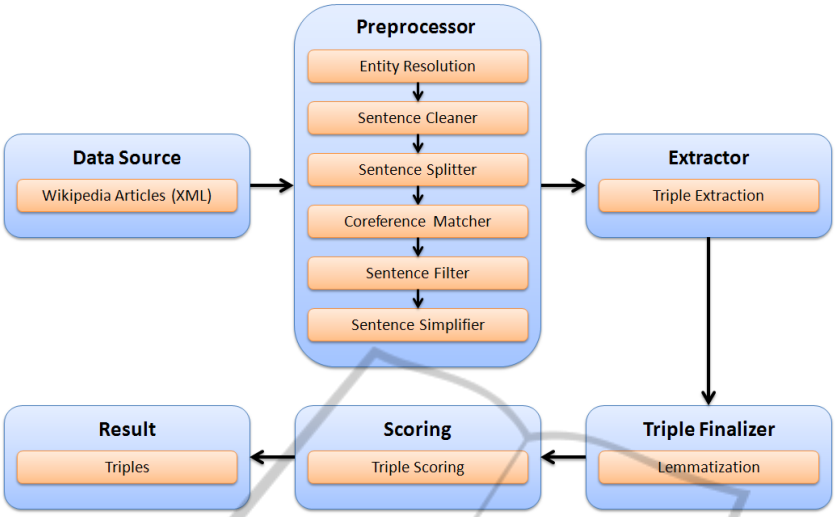

Figure 1: System Architecture

processor and the triple extractor. The preprocessor does most of the work; it prepares the natural language text in such a way that the extractor can easily extract the triples from the transformed input. Preprocessor and triple extractor will be described in more detail in sections 3 and 4.

2.1 Data Source

While YAGO (Suchanek et al., 2007) exploits the Wikipedia category system and KYLIN (Weld et al., 2009) uses Wikipedia infoboxes as their data source for extracting facts, we use the natural language text of Wikipedia articles because there are more facts embedded in the text. Furthermore, not all Wikipedia articles have infoboxes because the use of infoboxes in Wikipedia articles is optional. Thus, although the idea of providing articles' general information through Wikipedia infoboxes is favorable, many Wikipedia articles do not have infoboxes. On the other hand, most Wikipedia articles belong typically to one or more categories. Unfortunately, these categories are usually defined for quite general classes and are not as detailed as a classification which can be found in Wikipedia's natural language text. Another reason for choosing Wikipedia is that the articles are rich with markup around text fragments which can be utilized to extract facts with a higher precision.

3 PREPROCESSOR

The preprocessor is responsible for generating a set of sentences to be passed to our triple extractor. There are six steps, which are explained in the following subsections.

3.1 Entity Resolution

Entity resolution is a component in our system that utilizes metadata in order to synchronize multiple occurrences of the same entity expressed with different textual representations, e.g., "Albert Einstein" vs. "A. Einstein" . This component is invaluable towards reconciling different variants of an entity representation. For all cases, we always replace entities with their longest text representation in order to preserve information. In our current approach, we consider for reconciliation only entities corresponding to persons.

3.2 Sentence Cleaner

This component is responsible for cleaning sentences in articles from brackets and similar text fragments. We realized during the testing of our system that words inside brackets are often supplementary facts which give more details about the words occurring before the brackets. Moreover, it is also difficult for the triple extractor to understand linguistically the semantic relationship of the brackets and to correctly extract facts from them. Furthermore, we also extract metadata from the articles before we remove them completely from the texts. For each entity surrounded with metadata, we store its textual representation fragment and its URL, e.g., `<harry potter, http://en.wikipedia.org/wiki/Harry_Potter>`, into our metadata collection. This collection will be later used to remove unwanted sentences in the filtering process.

3.3 Sentence Splitter

After the text is filtered and cleaned, we split our input from paragraphs into a set of sentence chunks. Generally, one sentence may contain one or more triples. Sentence splitting is a crucial process towards ensuring that only qualitative sentences are fed to the system, this way ensuring triple extraction with higher precision. For this purpose, we rely on the sentence detection library provided by LingPipe (Alias-i, 2008).

3.4 Coreference Matcher

The need for coreferencing surfaces when multiple entities or pronoun words in a text refer to the same entity or have the same referent. In the first sentence of Wikipedia articles, an author typically introduces a person or an object with his full name (e.g., Albert Einstein). Then, in the following sentences the writer will begin using a substitute word (e.g., he, his, him, Einstein) for that person. Building the connection between these entities, e.g., *Albert Einstein* and *Einstein*, is what we refer to as coreferencing. Triples extracted from sentences with coreferences are fuzzy, because the context of the sentence is lost once the triple has been extracted. Even a human cannot understand the meaning of a triple such as `(he,received,Nobel Prize in Physics)` without a given context. Thus, we need to replace entities that have different representation forms into one single representation, i.e., full name. In order to do so, we apply the LingPipe's coreference resolution tool (Alias-i, 2008).

Although the purpose of the coreference matcher is similar with the entity resolution, which is to synchronize different associated entities into the same representation form, they work in a different way. In our approach, entity resolution only affects entities which are marked as links. Such links have been entered by humans manually and are highly accurate, therefore they can be used to correctly replace entities to their original representation (i.e., the title of the article describing the entity). On the other hand, the coreference matcher is able to catch entities which are not covered with metadata including pronoun words. However, its accuracy might not be as precise as entity resolution because LingPipe's coreferencer is a machine-based prediction system.

3.5 Sentence Filter

We remove invalid sentences which are caused either by the sentence splitting tool or by human errors. For example, sentences that do not start with a proper text or end with inappropriate punctuation symbols are ignored. Furthermore, we also exclude sentences which have a length over the defined threshold. This step is necessary to be imposed because with a deep parsing technique, long sentences demand high resources which ultimately slow down significantly the runtime of the whole system. Sentences with less than three words are removed as well, because a sentence at least needs to have a subject, a predicate, and an object to form a proper triple. After making sure that all sentences are valid by employing the above mentioned methods, we use the extracted metadata provided by the sentence cleaner to use those sentences with metadata only for triple extraction. Based on our observations, sentences with metadata are more likely to produce high quality triples.

3.6 Sentence Simplifier

An important step to improve the performance of the overall system is sentence simplification. The purpose of this step is to split complex sentences into sub-sentences. Wikipedia authors sometimes use rich stylish writing (e.g., list of items, dependent clauses, etc.) which makes some sentences become complex and hard to understand. Therefore, complex sentences need to be simplified in order to produce simple and clear triples as final result. We borrow the concept of sentence simplification processes from (Defazio, 2009).

We utilize the Stanford parser (Klein and Manning, 2003) to get the grammatical structure of sentences. The result of this parsing tool is a parse tree which will distinguish noun phrases (NP), verb phrases (VP), and group of words that belong together (e.g., dependent clauses (SBAR), prepositional phrases (PP), etc.) in the sentences. We use the generated parse tree to understand the correlation between words in a sentence and to split the sentences correctly. Nevertheless, parsing of natural language text is a challenge for any kind of NLP tool. Thus, not all generated parse trees are correct. Because our simplification process is really dependent on parse trees, the simplification results will be wrong if the provided parse trees are wrong. However, the overall results of this step are still decent and useful for the extraction process.

The sentence simplification has four steps:

Split Coordinating Conjunctions. Coordinating conjunctions are words like "and", "or", and "but" which are used in enumerations or to connect sub-sentences. An example from Albert Einstein's Wikipedia article is: '*his travels*

included Singapore, Ceylon, and Japan.' As we aim at supporting structured queries, we should extract three triples from this sentence and not only one. This would enable us to answer a question like 'Who travelled to Ceylon?' and not only the question 'Who travelled to Singapore, Ceylon, and Japan?'. Such simple enumerations could be still handled in a post-processing after the triples have been extracted, but we also want to handle more complex sentences like '*Welker and his department paved the way for microwave semiconductors and laser diodes*' (from the Wikipedia article about Heinrich Welker). The coreferencing step described above transforms 'his' into 'Welker's'. To extract the complete factual information from this sentence, we have to extract four triples from the sentence which correspond to the following statements:

- *Welker paved the way for microwave semiconductors.*
- *Welker paved the way for laser diodes.*
- *Welker's department paved the way for microwave semiconductors.*
- *Welker's department paved the way for laser diodes.*

Furthermore, we make also use of metadata in this step. If a text fragment is a link to another article, then this text fragment is not processed by this method.

Extract Dependent Clauses. In this simplification step, we will separate a dependent clause (SBAR in the terminology of the Stanford parser) from its main clause and create a new sentence which contains the SBAR clause and the subject from its main clause. There are two types of SBAR clauses: subordinate conjunctions (for adverbial clauses) and relative pronouns (for relative clauses) (Defazio, 2009). Our approach handles each type differently. The first type will be extracted into sub-sentence modifiers, while the second type will create a new sentence. For example, the sentence '*Albert Einstein also investigated the thermal properties of light which laid the foundation of the photon theory of light.*' contains a relative clause which is translated into '*Albert Einstein also investigated the thermal properties of light.*' and '*The thermal properties of light laid the foundation of the photon theory of light.*'

Extract Adjective Phrases. In this step, we extract adjective phrases from their main phrase. Adjective phrases generally appear in the middle of two phrases, i.e., a noun phrase and verb phrase, which are separated by comma. Adjective phrases could be a noun phrase (e.g., '*Planck's oldest son, Karl Weierstrass, was killed in action in 1916.*') or a verb phrase (e.g., '*Jacques Vergs, born 5 March 1925 in Siam, is a French lawyer...*'. We handle those two cases differently. For the noun phrases, we extract them from their main sentence and make them as the subject in the new sentence (e.g., we would get '*Planck's oldest son was killed...*' and '*Karl Weierstrass was killed...*'). Verb phrases will be used as the predicate (including the object) in the new sentence (e.g., '*Jacques Vergs born 5 March 1925 in Siam.*' and '*Jacques Vergs is a French lawyer...*').

Extract Secondary Verbs. Lastly, we extract all secondary verbs from the given sentences. These kinds of sentences contain two verb phrases with the same noun phrase. The second verb phrase is nested in the first verb phrase (e.g., '*Amy Lee Grant is an American singer-songwriter, best known for Amy Lee Grant's Christian music.*' would be translated into '*Amy Lee Grant is an American singer-songwriter.*' and '*Amy Lee Grant best known for her Christian music.*'

4 EXTRACTOR

Triple extractor is responsible for extracting facts and representing them in the form of triples. Triples are stored in the form of `(subject, predicate, object)`. We extract the triples by using the parse tree approach introduced in (Rusu et al., 2007). We use the Stanford parser (Klein and Manning, 2003) to generate the parse tree. The main idea of this approach is to utilize the parse tree as a helping tool to extract subjects, predicates, and objects from sentences, then combine them to produce a triple. In order to find the subject of a sentence, we have to search for it in the noun phrase (NP) tree. Furthermore, the predicate and the object of a sentence can be found in the verb phrase (VP) tree. By applying the simplification method of extracting dependent clauses and adjective phrases described above, we get these following results for the sentence '*Albert Einstein was a German-born theoretical physicist who developed the theory of general relativity, effecting a revolution in physics.*':

- `(Albert Einstein, was, German-born theoretical physicist)`
- `(Albert Einstein, developed, theory of general relativity)`
- `(Albert Einstein, effecting, revolution in physics)`

4.1 Triple Finalizer

The triple finalizer converts the predicate of the triple results into their lemmatized form. The lemmatized form is a base form of a verb, for example, the word *receiving* or *received* will be changed to the word *receive* as its lemma form. Lemma results prevent ambiguity and is useful for queries. All triples produced by this step are the final triples of our system. For this component, our system uses a lemmatizer provided by the Stanford Natural Language Processing Group (Toutanova et al., 2003).

4.2 Scoring

We assign lower scores for triples which do not contain metadata. Triples with metadata deserve a higher score because the metadata ensure the quality of the sentence and also the information which they surround. Furthermore, the completeness of a triple (i.e. subject, predicate, and object) and the entity type of the subject (i.e., PERSON, LOCATION, and ORGANIZATION) also determine the triple scores. The entity types are assigned during the coreferencing step by the LingPipe's Named Entity Recognition (Alias-i, 2008). The following formula is used to compute the score of the triples:

$$score = \frac{metadata + completeness + entity}{3}$$

The score is a hint on the quality of the extracted triple; it could be used in further processing steps (e.g., in ranking results of a structured query). However, this was not the focus of our work, and we are aware that a more sophisticated scoring mechanism might be required which includes more factors that indicate the quality of the extracted triple. This is an issue which we will address in future work.

5 EVALUATION

The evaluation is divided into two parts: a user-based evaluation (sections 5.1 and 5.2) and a system-based evaluation which compares our system with ReVerb (Etzioni et al., 2011) (section 5.3). ReVerb is a recent system for open information extraction and has shown good performance in its evaluation.

5.1 User-based Evaluation: Setup and Dataset

The goal of the evaluation is to verify the correctness of the extracted triples as we aim at extracting high quality triples. As there is no golden standard for the task of triple extraction over Wikipedia, we had to take a sample of documents from Wikipedia, apply the FactRunner method, and rate the correctness of the extracted triples manually. In order to avoid a very subjective rating of correctness of the extracted triples, several users were involved in the evaluation. A triple was considered as correct when all users agreed on the correctness of that triple.

The evaluation was done using a web-based interface. For each sentence, the list of extracted triples was presented. For each triple, the user could state whether the triple is correct or not. As stated before, we consider only correctness and not completeness; therefore, we did not ask the user for other triples that could have been extracted from the sentence.

We conducted our evaluation by using two categories of article in Wikipedia: *American Actors* and *American Physicists*. Table 1 summarizes some statistics about the dataset and the extracted triples.

5.2 User-based Evaluation: Results

For the user-based evaluation, we picked 15 articles from each category as the predefined documents that we presented to the users. The overview of this evaluation is shown in Figure 2. The y-axis of the figure shows the number of triples extracted (total, correct, and incorrect). The average precision of the used-based evaluation system is 77%. It shows promising result where 6 from 30 predefined articles have precision equal or more than 90%. We believe the complex process of the sentence preprocessing, especially the sentence filter and sentence simplifier component, are the main reason for this high value. Nevertheless, three articles from our datasets obtained precision below than 60%. After we looked into our result, we found that the main reason for this low precision is because the subjects of the triples are not clear (e.g., he, she, her, these, their, etc.). The reason for this is that the coreference matcher failed to change pronoun words into the corresponding entities in the subject of the triples. Thus, the users are confused and evaluated those triples as incorrect triples.

5.3 System-based Evaluation

Table 2 summarizes the results of the comparison of our system with ReVerb. The dataset is the same as for the user-based evaluation. The numbers show how many triples we are able to extract for the dataset wrt. ReVerb. Thus, it is indicator for the recall of the system; however, as it is very time consuming to define a reference result for such a large dataset,

Table 1: Statistics of the dataset and the extracted triples.

	American Actors	American Physicists
# Articles	2,483	1,298
Total #Triples	51,088	26,967
Triples with Full Score	39,313	19,975
Contains Entity	46,735	23,394
Contains Metadata	42,140	22,524
Triple Completeness	48,510	25,587

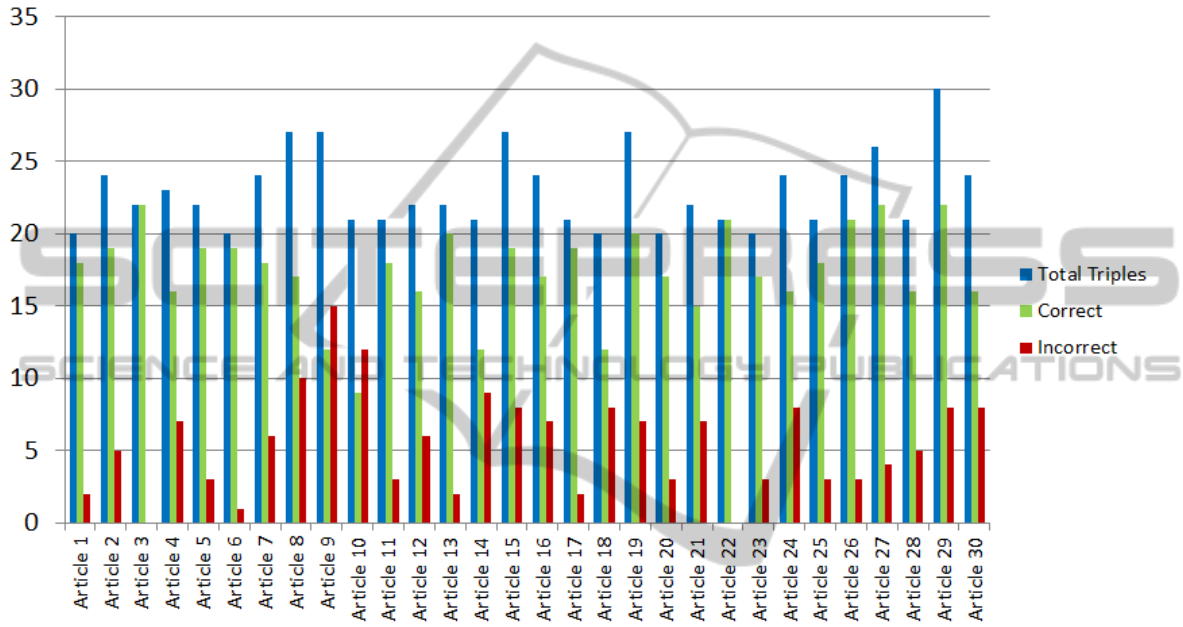

Figure 2: Overview of user-based evaluation.

an exact computation of the recall value is not possible. Although we used the same datasets for both systems, our system considered only sentences which have metadata and are shorter than 200 characters. Thus, the number of sentences considered by our system is less than in ReVerb. The evaluation shows that both systems are able to extract multiple facts from one sentence. In average, FactRunner is able to extract 1.81 triple/sentence and ReVerb is able to extract 1.3 triple/sentence. Thus, our system is able to extract more triples with fewer sentences. The reason behind this result is most likely that FactRunner uses a deep-parsing technique in order to extract its triples. However, the cost of this technique is a slow processing speed (about 7-8 times slower than ReVerb on the same machine).

5.4 Performance

We tested our system on a PC with Windows XP 64bit, Intel Core2 Quad CPU Q9550 (2.83GHz), and 8 GB RAM. The system is implemented in Java and uses the aforementioned libraries (LingPipe, Stanford Parser, etc.). Due to the NLP techniques which analyze the given texts in detail, the system needs about 3 seconds to process one document. The runtimes for the American actors dataset were about 128 minutes, and about 64 minutes for the American physicists dataset. This results in an average time of about 20 documents per minute, or 3 seconds per document. In contrast, ReVerb achieved a value of about 160 documents per minute. However, we must emphasize that performance was not yet the focus of our system. For example, we do not make use of the multicore CPU by using parallel threads. Parallization or distribution in a cluster of the application would be easily possible, because each document can be processed independently.

Figure 3 shows the distribution of the processing time for each step of the FactRunner system. Most of the time is spent in sentence simplification because it requires the analysis of the natural language text using an NLP tool. Nevertheless, the additional effort pays off as the extraction is simplified to a great ex-

Table 2: Comparison of FactRunner and ReVerb.

	FactRunner		ReVerb	
	American Actor	American Physicist	American Actor	American Physicist
# Articles	2,483	1,298	2,483	1,298
# Considered Sentences	27,489	15,371	40,760	25,625
# Extracted Triples	51,088	26,967	52,817	33,620
Triples/Sentence	1.86	1.75	1.30	1.31

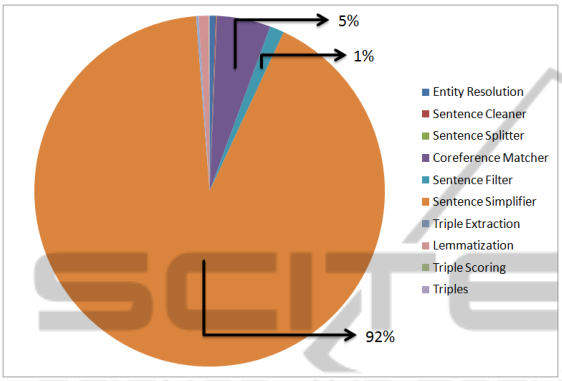

Figure 3: Distribution of processing time.

tend. In a previous version of the system, the extractor required much more resources because it had to apply the NLP techniques on more complex sentences.

6 RELATED WORK

There are two major bodies of work in the area of information extraction (IE). The first body of work relies on pattern-based techniques, whereas the second relies on natural language processing methods.

6.1 Pattern-based Techniques

The pattern-based techniques, as the term implies, are techniques used to identify patterns in order to extract information in the target corpus. Patterns are skeletal frames of text fragments which can be used to extract relations in a given text corpus. Such pattern-based extraction systems are able to extract relations if a given pattern is matched with an existing relation in the document collection (Blohm, 2010). Before the extraction process, pattern-based techniques generally need to prepare a limited number of patterns to be taken into account in the extraction process. Pattern-based approaches perform well when considering precision, but they cannot capture information that does not use the predefined patterns. KnowItAll (Etzioni et al., 2004), Snowball (Agichtein and Gravano, 2000), DIPRE (Brin, 1999), WOE (Wu and Weld, 2010), and YAGO (Suchanek et al., 2007) are all successful existing systems which use this extraction paradigm. The YAGO ontology uses Wikipedia as their source, as we do it in our approach. YAGO combines the Wikipedia category system with the WordNet lexical ontology in order to extract information which in turn creates a larger ontology.

6.2 Methods based on Natural Language Processing

Our approach is similar with the other direction of extraction technique, i.e., natural language processing (NLP). This technique relies on NLP tools which focus on analyzing natural language text. The NLP approach is able to handle an unbounded number of relations because it does *pattern recognition* and not *pattern matching*. An NLP-based approach can capture a large amount of information from many sources, for instance the World Wide Web. Nevertheless, natural language processing is a complex and ambiguous process. To the best of our knowledge, there are no NLP tools that can perfectly understand natural language text, thus resulting errors cannot be avoided. Our approach is a combination of NLP techniques, as well as proliferation of the already existing metadata in the text, in order to perform a precise relation extraction from English natural text.

TextRunner (Banko et al., 2007) is a good example showing the benefits of NLP-based techniques. It is assembled based on the idea of Open Information Extraction (OIE) paradigm. OIE is a newly developed approach of information extraction system that could provide relation independence and high scalability, handling a large corpus such as the Web corpus. Similar to our approach, TextRunner is able to extract information from a vast variety of relations located in a given corpus with only a single pass. However, our approach is mainly targeting Wikipedia articles and its metadata; not the World Wide Web. In short, our goal is to further enhance the extraction process by utilizing the already existing metadata available in the Wikipedia collections.

Another good representative of the NLP-based approach is KYLIN (Weld et al., 2009). KYLIN uses

Wikipedia infoboxes as a training dataset in order to extract information from Wikipedia and the Web. The result of this approach is a system that automatically creates new infoboxes for articles which do not have one, as well as completing infoboxes with missing attributes. Unfortunately, not all Wikipedia articles have infoboxes. Infoboxes also suffer from several problems, which are: incompleteness, inconsistency, schema drift, type free system, irregular lists, and flattened categories (Wu and Weld, 2007). In contrast to KYLIN, our approach uses the existing metadata available in almost every Wikipedia article. Our approach treats each sentence with metadata individually, thus we are not relying on any specific limited resources for training datasets. Furthermore, although KYLIN can learn to extract values for any attributes, their attribute sets are limited to those attributes occurring in Wikipedia infoboxes only.

7 CONCLUSIONS AND OUTLOOK

In this paper, we have presented a novel approach for extracting facts from the Wikipedia collection. Our approach utilizes metadata as a resource to indicate important sentences in Wikipedia documents. We also have implemented techniques to resolve ambiguous entities (i.e., persons) in those sentences. Furthermore, we simplify complex sentences in order to produce better triples (e.g., one triple contains only one fact which corresponds to a sentence) as the final result of our system. The simplification also improves the runtime of our system. We also apply lemmatization of the triples' predicates to have a uniform representation, which simplifies querying and browsing of the extracted information. The evaluation has shown that we can achieve a high precision of about 75%.

For future work, we have several ideas to improve the current version of the system. The first idea is to improve our scoring component. Triples extracted from the first sentences of a document could get higher scores as these sentences usually contain very clear facts. Furthermore, the transformation which we have applied during the preprocessing step should be also taken into account in the score of a triple (e.g., a triple with a subject derived from coreferencing is less certain). Another important issue is the consolidation and normalization of the triples. We already apply lemmatization and entity resolution, but further consolidation according to the semantics of predicates would be helpful for querying. Nevertheless, we think that our proposed system provides a good basis for extracting high quality triples from Wikipedia.

ACKNOWLEDGEMENTS

This work has been supported by the German Academic Exchange Service (DAAD, http://www.daad.org) and by the DFG Research Cluster on Ultra High-Speed Mobile Information and Communication (UMIC, http://www.umic.rwth-aachen.de).

REFERENCES

Agichtein, E. and Gravano, L. (2000). Snowball: Extracting relations from large plain-text collections. In *Proc. 5th ACM Intl. Conf. on Digital Libraries*, pages 85–94.

Alias-i (2008). LingPipe 4.1.0.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction from the web. In Veloso, M. M., editor, *Proc. 20th Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2670–2676, Hyderabad, India.

Blohm, S. (2010). *Large-scale pattern-based information extraction from the world wide web*. PhD thesis, Karlsruhe Institute for Technology (KIT).

Brin, S. (1999). Extracting patterns and relations from the world wide web. In Atzeni, P., Mendelzon, A. O., and Mecca, G., editors, *Proc. Intl. Workshop on The World Wide Web and Databases (WebDB)*, volume 1590 of *Lecture Notes in Computer Science*, pages 172–183. Springer.

Burton-Jones, A., Storey, V. C., Sugumaran, V., and Purao, S. (2003). A heuristic-based methodology for semantic augmentation of user queries on the web. In *Proc. 22nd Intl. Conf. on Conceptual Modeling (ER)*, volume 2813 of *LNCS*, pages 476–489.

Defazio, A. (2009). Natural language question answering over triple knowledge bases. Master's thesis, Australian National University.

Dong, H., Hussain, F., and Chang, E. (2008). A survey in semantic search technologies. In *Proc. 2nd Intl. Conf. on Digital Ecosystems and Technologies (DEST)*, pages 403–408. IEEE.

Etzioni, O., Cafarella, M. J., Downey, D., Kok, S., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. (2004). Web-scale information extraction in knowitall: (preliminary results). In *Proc. WWW*, pages 100–110.

Etzioni, O., Fader, A., Christensen, J., Soderland, S., and Mausam (2011). Open information extraction: The second generation. In *Proc. IJCAI*, pages 3–10, Barcelona, Spain.

Halevy, A. Y., Etzioni, O., Doan, A., Ives, Z. G., Madhavan, J., McDowell, L., and Tatarinov, I. (2003). Crossing the structure chasm. In *Proc. 1st Biennal Conference on Innovative Data Systems Research (CIDR)*, Asilomar, CA, USA.

Heath, T. and Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers.

Hoffart, J., Suchanek, F. M., Berberich, K., Lewis-Kelham, E., de Melo, G., and Weikum, G. (2011). Yago2: Exploring and querying world knowledge in time, space, context, and many languages. In *Proc. WWW (Companion Volume)*, pages 229–232.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In Hinrichs, E. W. and Roth, D., editors, *Proc. 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 423–430, Sapporo, Japan.

Mangold, C. (2007). A survey and classification of semantic search approaches. *International Journal of Metadata, Semantics and Ontologies*, 2(1):23–34.

Rusu, D., Dali, L., Fortuna, B., Grobelnik, M., and Mladenic, D. (2007). Triplet extraction from sentences. In *Proc. 10th Intl. Multiconference on Information Society*, volume A, pages 218–222.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A Core of Semantic Knowledge. In *Proc. WWW*.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. Intl. Conf. of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 173–180, Stroudsburg, PA, USA.

Weld, D. S., Hoffmann, R., and Wu, F. (2009). Using Wikipedia to bootstrap open information extraction. *SIGMOD Record*, 37(4):62–68.

Wu, F. and Weld, D. S. (2007). Autonomously semantifying wikipedia. In Silva, M. J., Laender, A. H. F., Baeza-Yates, R. A., McGuinness, D. L., Olstad, B., Olsen, Ø. H., and Falcão, A. O., editors, *Proc. 16th Confón Information and Knowledge Management (CIKM)*, pages 41–50, Lisbon, Portugal. ACM.

Wu, F. and Weld, D. S. (2010). Open information extraction using wikipedia. In Hajic, J., Carberry, S., and Clark, S., editors, *Proc. 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 118–127, Uppsala, Sweden.