View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Digital Library of the Czech Technical University in Prague

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER'S THESIS

Title:	Fact extraction from Wikipedia article texts
Student:	Bc. Jakub Trhlík
Supervisor:	Ing. Milan Dojčinovski, Ph.D.
Study Programme:	Informatics
Study Branch:	Web and Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of winter semester 2020/21

Instructions

DBpedia is an open and free knowledge graph which provides structured information extracted from Wikipedia. Currently, the information in DBpedia has been extracted from semi-structured sources such as Wikipedia infoboxes. The ultimate goal of the thesis is to enrich the DBpedia knowledge graph with information (facts) extracted from unstructured sources - Wikipedia article texts.
Guidelines:
- Analyze existing methods for fact extraction from texts.
- Get familiar with the DBpedia NIF dataset, which provides the underlying Wikipedia article content.
- Implement and adapt selected fact extraction methods on Wikipedia article texts.
- Apply the implemented methods and extract facts from Wikipedia article texts.
- Evaluate the quality of the results.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague March 8, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Fact extraction from Wikipedia article texts

Bc. Jakub Trhlík

Department of Software Engineering
Supervisor: Ing. Milan Dojčinovski, Ph.D.

January 9, 2020

Acknowledgements

Thank to my family, friends and Ing. Milan Dojčinovski for support.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on January 9, 2020 .....

Czech Technical University in Prague
Faculty of Information Technology
© 2020 Jakub Trhlík. All rights reserved.
This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Trhlík, Jakub. *Fact extraction from Wikipedia article texts.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

Wikipedia je skvělý zdroj informací, v současné době z ní ale nejsou textové informace extrahovány do strojově čitelného formátu. V této práci využíváme DBpedia NIF dataset, představující strukturu stránek Wikipedie, pro cílenou extrakci faktů. Dataset je analyzován, obohacen o odkazy pomocí několika metod a poté připraven na extrakci faktů. V této práci je zkoumáno, implementováno a testováno několik metod extrakce faktů na vybraných vztazích. Experimenty popisují přesnost a použitelnost vybraných a implementovaných metod. Extrahované vztahy jsou vyhodnoceny a odeslány k přidání do DBpedie.

Klíčová slova DBpedia, extrakce vztahů, klasifikace, porozumění textu, strojové učení, hluboké učení, NLP

Abstract

Wikipedia is great source of information, currently its text information has not been extracted into fully machine-readable format. In this thesis, we use DBpedia NIF dataset, representing Wikipedia page structure, for targeted fact extraction. The dataset is parsed, enriched by links using several methods and

then prepared for fact extraction. In this thesis multiple methods of fact extraction are researched, implemented and tested on selected relations. Experiments describe accuracy and viability of selected and implemented methods. Extracted relations are evaluated and submitted for addition to the DBpedia database.

Keywords DBpedia, relation extraction, classification, natural language understanding, machine learning, deep learning, NLP

Contents

List of Figures

List of Tables

Introduction

Motivation

There has been a great development in the area of text extraction and understanding. It is being used in intelligent assistants, searches and related software. The decreasing limitations of computing power and newly found methods allowed for uprise of these technologies, but also unveiled limits in other areas. Knowledge bases play one of the most important roles. The ability for today's techniques to understand human written texts in broader context to create and update knowledge bases is still at the beginning.

One of the biggest knowledge bases is DBpedia [3]. DBpedia is big knowledge graph based on data in Wikipedia's info-boxes. DBpedia's data are strictly in line with Linked Data principles using open standards[4].

Goals of the thesis

The main goal of this thesis is to enrich DBpedia knowledge base, by extracting facts directly from Wikipedia texts, using modern machine learning techniques.

Subtasks include:

- parsing the NIF dataset,
- enriching dataset with additional links,
- preprocessing dataset for machine learning and fact extraction,
- realization of relation extraction techniques,
- experiments using implemented techniques,
- evaluation of implemented techniques,

- extracting relations.

This thesis is restricted to English language and all resources, data sets, statistics and tools are going to be used and described for English versions.

Definitions used

By referring to DBpedia in this thesis, it is meant DBpedia Knowledge Base. Referring to the DBpedia organisation is going to be explicitly noted.

Definition 0.0.1. DBpedia in this thesis, is meant to be DBpedia Knowledge Base. Referring to the DBpedia organisation is explicitly noted.

Background and Related Work

In this chapter, topics immediately related to the assignment of the thesis are researched. We focus on Knowledge bases (especially DBpedia), their structure and how they can be leveraged for the assignment. We research existing methods of fact extraction and types of fact extraction. Machine learning is researched and in which ways can be used for fact extraction.

Knowledge bases

Definition 0.0.2. Knowledge base (KB) is a technology used to store complex structured and unstructured information used by a computer system[5].

The term knowledge base as defined above is too abstract for purposes of this thesis and is therefore further specified.

Definition 0.0.3. Graph knowledge base (GKB) or **Knowledge graph** is a graph-based database represented as entities and relations between them. It is used to store complex structured information.

Example of graph knowledge base can be found at 0.1.

The reason for this definition is for the entity and relations between entities to exist in the database. Further on, graph knowledge base can be referred to as Knowledge bases or KB.

Properties of Graph knowledge bases:

- public availability,
- degree of specialisation,
- language availability,
- credibility and precision,
- ontology/no ontology,

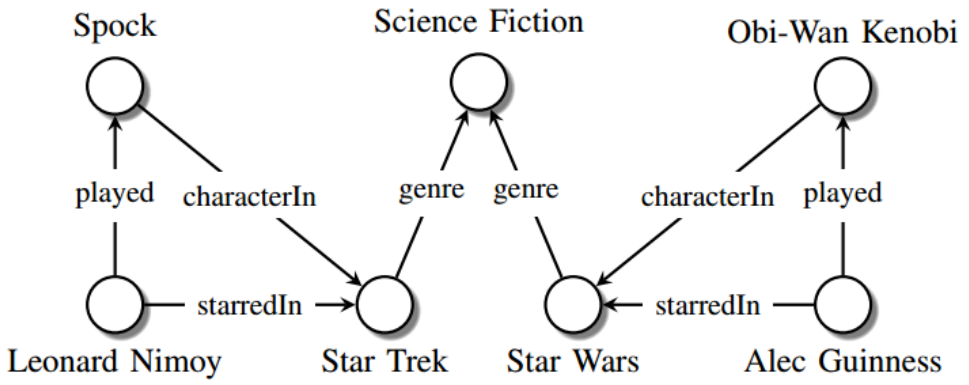

Figure 0.1: Example of graph knowledge base structure from From Multi-Relational Link Prediction to Automated Knowledge Graph Construction [1]

- number of Relations,
- number of Entities.

Definition 0.0.4. Ontology In computer science and information science, an ontology encompasses a representation, formal naming and definition of the categories, properties and relations between the concepts, data and entities that substantiate one, many or all domains of discourse[6].

Knowledge Base structure can be based on defined structure or hierarchy principles between entities and their relations called ontologies or can be structurally inconsistent.

Wikipedia

Wikipedia, owned by a nonprofit organisation, is one of most popular websites as of 2019. As it is a website, consisting of pages with unique URL, it is a graph knowledge based website, where each node is page with full of unstructured information.

Wikidata

Wikidata is a wikimedia project. It is open knowledge base that can be read by machine. Wikidata acts as central storage for the structured data of its Wikimedia sister projects: Wikipedia, Wikivoyage, Wiktionary, Wikisource, and others[7]. The most occurred types of items in Wikidata are human, taxon, administrative territorial entity, architectural structure, occurrence, chemical compound. It is necessary to state, that 43% of Wikidata consists of scholarly articles, that is main reason why Wikidata is often overrated.

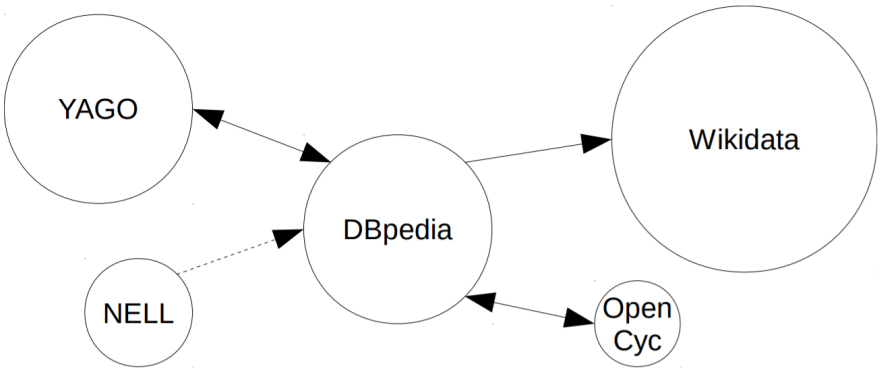

Figure 0.2: DBpedia and their interlinks. The size of the circles reflects the number of instances [2]

Structure

Wikidata consists of 3 types of entities: **properties**, **items** and **queries**. **Item** is an concept, object or topic, each item is identified with unique identifier starting with letter Q known as QID. Item consists of identifier QID, labels, aliases, descriptions, statements and site links. **Property** is the descriptor of data value related to an item. In association of linked data, the property is type of a triplet's predicate. Statement is an key/value pair such as occupation property for item. Statement links item page via property to value. In association of linked data, the statement represents a fact or a triplet together with item, property and value.

Freebase

Freebase was a knowledge base that consisted of structured data created by its community members. The goal of Freebase was to create globally accessible resource of everything, that would allow anyone to access common information automatically. Metaweb, creator of Freebase, was acquired by Google in 2010. Data of Freebase continues to exist within other knowledge bases such as DBpedia and Google's Knowledge graph. Last Freebase version is still available as Data Dump provided by Google[8].

YAGO

YAGO (Yet Another Great Ontology) is an open source knowledge base developed at the Max Planck Institute for Computer Science in Saarbrücken. It is automatically extracted from Wikipedia and other sources[9].

Other

In latest years, most of the big technological companies, that work with information are building a knowledge graph. Company-owned knowledge graphs, like the Google Knowledge Graph, Yahoo's Knowledge Graph, Microsoft's Satori, and Facebook's Knowledge Graph are being used for the company's services. However, those graphs are not publicly available, and can not be analyzed in-depth.

Other graphs worth mentioning: FOAF[10], GeoNames, UMBEL

DBpedia

In this section DBpedia is analysed. Based on the thesis task, DBpedia NIF dataset is used for relation extraction and so is analyzed more in-depth than other knowledge bases.

About

Definition from DBpedia web page[3]: DBpedia is a crowd-sourced community effort to extract structured content from the information created in various Wikimedia projects. This structured information resembles an open knowledge graph (OKG) which is available for everyone on the Web.

The project of DBpedia as of now is joined project of Leipzig University, Free University of Berlin, University of Mannheim, Hasso Plattner Institute and company OpenLink Software.

DBpedia is one of the key project of the decentralized Linked Data, or Linked Web as for now. It uses a resource URL as resource URI. This URI is derived from wikipedia URL as an unique entity identifier.

The DBpedia organization release regularly several official dataset dumps in NIF and TTL formats. Recently these datasets started include also unstructured wikipedia texts. Other way how to obtain these data sets is to use DBpedia Extractors for Wikipedia resp. Wikimedia available at GitHub.

As of writing this thesis DBpedia web page and download page has not been recently updated, some of the pages like dumps or liveupdates are no longer available.

DBpedia as of now is used to help organize and structure content on many platforms. Samsung includes DBpedia data in its knowledge platform, BBC uses DBpedia, Faviki uses DBpedia for tagging.

It is possible to query DBpedia using SPARQL querying language and provided API. Querying can be also done online using Virtuoso service [11].

Properties

The English version of the DBpedia knowledge base describes 4.58 million entries, out of which 4.22 million are classified in a consistent ontology, includ-

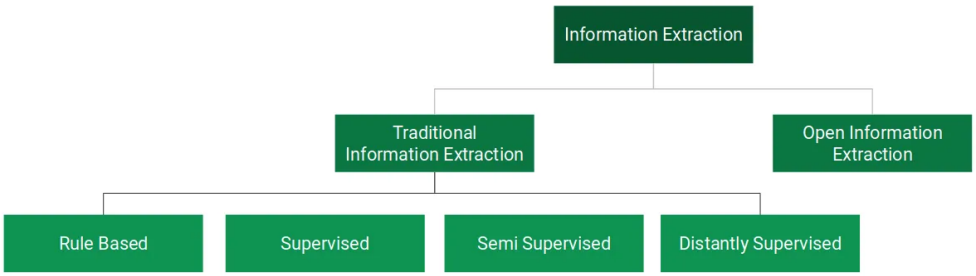

Figure 0.3: Types of relation extraction

ing 1.445.000 persons, 735.000 places (including 478.000 populated places), 411.000 creative works (including 123.000 music albums, 87.000 films and 19.000 video games), 241.000 organizations (including 58.000 companies and 49.000 educational institutions), 251.000 species and 6.000 diseases[3].

Structure

The current version of ontology is available at DBpedia mappings[6]. The DBpedia ontology has been created manually and is not automatically extensible. The ontology is derived from Freebase ontology and existing Wikipedia infobox properties. Ontology has 685 classes in a directed-acyclic graph described by 2795 properties. Ontology has mappings to the schema.org, which is another Linked web project.

Relation extraction

Task of relation extraction is defined as extraction of relational triples such as (Aspirin, treats, pain) from natural language text. Relation extraction is one of main tasks of information extraction and information understanding. Allows knowledge base creation and better understanding of the text.

There are five main classes of algorithms for relation extraction:

Hand-written patterns

The easiest and still widely used technique for relation extraction is hand-written patterns. Is based on lexico-syntactic patterns.

Hand-written patterns were first used by Hearst at 1992.

For example for hyponym relation Hearst suggests [12]:

pattern

$$NP_1\{NP_2...(and\|or)NP_i\}, i \geq 1 \tag{1}$$

that implies

$$\forall NP_i, i \geq 1, hyponym(NP_i, NP_0) \tag{2}$$

Hand-build patterns have high-precision and can be used in almost any domain. But patterns have often low recall and it takes big amount of work to create them.

One way to make this work easier is to use machine learning or some algorithm to recognise these patterns or group of patterns automatically.

Supervised machine learning

Supervised machine learning approach requires hand-annotated data. Training corpus is hand-annotated with the relations and entities. The training corpus is then used to train classifiers. Resulting model is then applied to annotate an unseen set of texts.

General algorithm for finding relations:

```
function findRelations(words)
    relations=nil
    entities=findPairsOfEntities(words)
    forall entity pairs <e1, e2> in entities do
        if findIfIsRelated(e1, e2)
            relations+=classifyRelation(e1, e2)
    returns relations
```

Features

The most important step for feature-based classifiers is to identify useful features.

Embeddings

Embedding is method of replacing or adding information to the data itself. The most popular is Word embedding, where words can be replaced by their hypernyms or categories to generalise the data without loosing too much information.

- Entity hypernym, or entity types and their combination, can be focused at Named entities or entities as general,
- part of speech embeddings,
- distance to each entity embedding,
- word to vector embeddings.

Word features

- Combination of important words, targeted entities and verb in order,
- bag of words or bigrams,
- distance of targeted Entities including their order in sentence.

Syntactic structure

- Syntactic trees,
- path traversed through the tree in getting from one entity to the other.

Classifiers

- Nearest Neighbour,
- naive Bayes,
- decision Trees,
- linear Regression,
- support vector machines (SVM),
- neural Networks.

Semi-supervised via bootstrapping

Unfortunately, supervised machine learning needs a lot of data to train reliable classifier. One of the ways to get enough data is via method of bootstrapping. Bootstrapping uses initial seed tuples and then finding sentences that contains both entities.

Bootstrapping methods

Dual Iterative Pattern Relation Extraction – DIPRE algorithm by Sergey Brin [13]

- Start with a small sample R0 of the target relation. This sample is given by the user and can be very small.
- Then, find all occurrences of tuples of R0 in Data. Along with the tuple found, keep the context of every occurrence.
- Generate patterns based on the set of occurrences. This procedure must generate patterns for sets of occurrences with similar context. The patterns need to have a low error rate.

- Apply the patterns to data, to get a new set of relation pairs.
- Return to step 2, and iterate until convergence criteria is reached

Snowball algorithm by Eugene Agichtein and Luis Gravano [14]

- Start with a small sample R0 of the target relation. This sample is given by the user and can be very small, for example 5-10 seeds.
- Group found instances with similar prefix, middle and suffix, extract patterns based on most.
- Require that X and Y be named entities and compute confidence for each pattern.
- Apply the patterns to data to get a new set of relation pairs.
- Return to step 2, and iterate until convergence criteria is reached.

From found data, we can train a classifier which can then classify and find new relations.

Semi-supervised via distant supervision

Other way to get at labeled data is with distant supervision method.

Distant supervision uses existing information about relations between entities to create classifier. Distant supervision uses large databases to acquire big number of examples.

Using these examples for labelling usually creates also a lot of noisy inaccurate patterns. Combining this training data in supervised classifier together with labelled counter-examples may filter the noise and create accurate classifier.

One of recent examples is work of Mintz et al. (2009)[15]. He combines bootstrapping and distant supervision with supervised learning. In this work, 800 000 Wikipedia articles was used to extract all sentences that have 2 named entities matching the searched tuple. These found sentences were then used in supervised classifier.

Unsupervised

Unsupervised relation extraction is often called Open information extraction or Open IE. Open IE is about extracting relations, when there is no labeled training data and not even any list of relations.

Linked data

Linked data is a concept or vision, of interlinking the World Wide Web into one big query-able database. The main goal is for computers to be able to read it, search it and possibly learn from it.

Principles

- Use URIs to name (identify) things.
- Use HTTP URIs so that these things can be looked up (interpreted, "dereferenced").
- Provide useful information about what a name identifies when it's looked up, using open standards such as RDF, SPARQL, etc.
- Refer to other things using their HTTP URI-based names when publishing data on the Web.

Machine learning

Machine learning in latest years revolutionised many fields or interest. There has been big improvements especially in natural language understanding in combination with word vector models.

Machine learning is the science of algorithms and statistical models ability to learn from data.

There are three basic types of machine learning. Supervised, Unsupervised and Reinforcement learning depending upon the nature of the data it receives. Sometimes also Semi-supervised machine learning is considered as its own category, but differs from Supervised machine learning by the method, how the training dataset is created.

In this thesis we focus on supervised learning because reinforcement learning and unsupervised learning is not considered well suited for relation classification.

Supervised learning

Supervised machine learning set of algorithms builds a mathematical model. This model is created based on given training data, which consist of inputs and corresponding outputs. The goal is to create model which predicts output on given input. Each training example is represented by an array of vectors, called feature vector. The model is then trained through iterative optimisation[16].

Supervised learning can be used to solve 2 types of tasks.

- Regression: predicting a continuous numerical value.

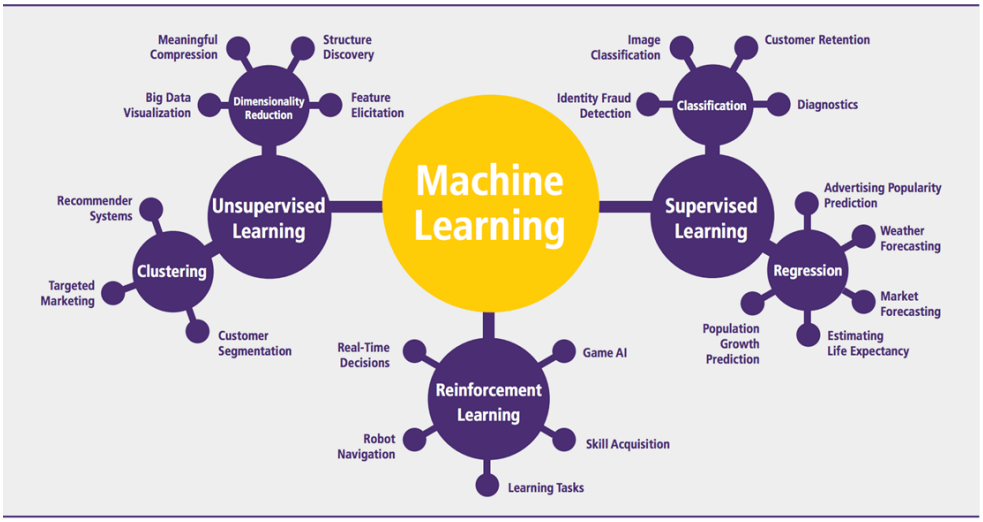

Figure 0.4: There are three basic types of machine learning

- Classification: way of classifying outcomes into different classes.

Because our problem is defined as categorising relations, the machine learning algorithms we consider in this thesis will be used for classification.

In this thesis selected best suited machine learning algorithms are used and compared.

Logistic Regression

Logistic regression is similar to linear regression. The difference between linear regression and logistic regression is, what is the algorithm used for. Linear regression predicts continuous values, but logistic regression is used for classification task.

As linear regression, logistic regression also uses linear equation inside with independent predictors to predict a continuous value, but output algorithm predicts the final value.

There are several output equations, that can be used for this task, most used equation is simple sigmoid function.

Logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1. The logistic function is defined as:

The relationship between the outcome and the given features are defined by linear equation:

Artificial neural network

The idea of Artificial neural network is inspired by how the biological neural networks in animal brains work.

$$sig(z) = \frac{1}{1 + e^{-z}}$$

Figure 0.5: sigmoid function

(3)

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 ...$$

Figure 0.6: linear equation

(4)

The development of Artificial neural networks started 1950s, in 1958 perceptron based network was invented. The function of backpropagation was then developed in 1960. Only in recent years the computing performance got onto level where Artificial neural networks are often beating linear classifier and support vector machine approaches.

Artificial neural network (ANN) and derived technologies are most promising machine learning algorithm for Natural Language Understanding and relation classification.

A good definition of ANN is in the book Neural networks and learning machines by / Simon Haykin [17]. He describes ANN as a massive graph of nodes, simple processing units, which can store information in its connections.

Definition 0.0.5. Artificial neural network is a directed graph consisting of nodes with interconnecting synaptic and activation links and is characterised by four properties:

- Each neuron is represented by a set of linear synaptic links, an externally applied bias, and a possibly nonlinear activation link. The bias is represented by a synaptic link connected to an input fixed at +1.
- The synaptic links of a neuron weight their respective input signals.
- The weighted sum of the input signals defines the induced local field of the neuron in question.
- The activation link squashes the induced local field of the neuron to produce an output.

Neuron can be described mathematically as:

$$y_k = \phi(v_k) \tag{5}$$

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{6}$$

Activation function

Hyperbolic tangent sigmoid activation function is symmetric bipolar activation function, defined by

$$\phi(v) = \frac{2}{(1 + e^{-2v})} - 1 \tag{7}$$

The ANNs work best if they are dealing with non-linear dependence between the inputs and outputs.

Backpropagation

Backpropagation is method for adjusting the connection weights using gradient descent. The adjustion is based on errors found during learning from the training data.

Perceptron

Perceptron or One-layer Neural network is type of ANN using activation function called Heaviside step. This basic concept was introduced by Rosenblatt in 1958. However because Heaviside step is unit step function, it is very limiting, so today hyperbolic tangent or logistic sigmoid functions are being used instead.

Autoencoder

Autoencoder is also a type of neural network. Purpose of autoencoder is to learn a representation for a set of data and encode it as a vector.

Feature vector

The prediction of classification is decided based on feature vector of measurable properties classified instance. Each of the properties is called a feature. Features are usually determined manually for the model to get the best results.

Deep learning

Deep learning is a subset of machine learning algorithms based on neural networks. Deep learning uses multiple layers of neural networks into layered structure, where information is passed between the layers.

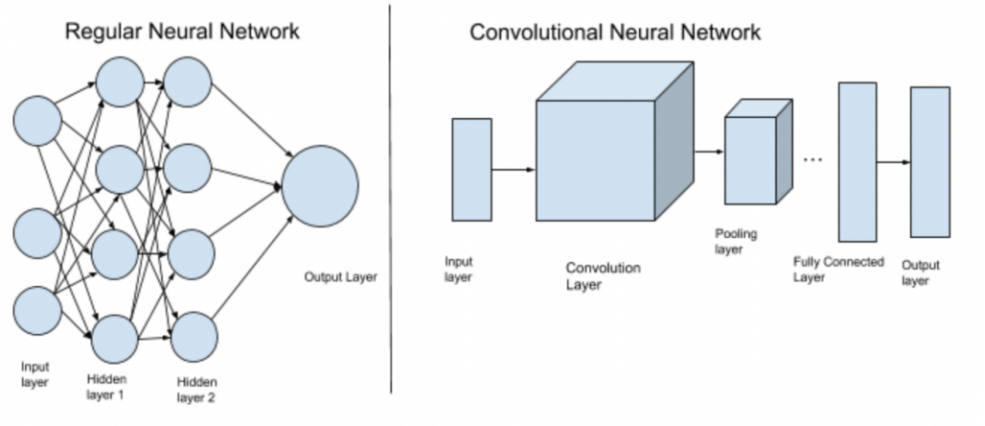

Figure 0.7: CNN schema

The difference between machine learning and deep learning is, that deep learning does the feature extraction also, machine learning doesn't have to, sometimes the feature extraction is done by human.

Deep learning algorithms are good especially in situations where the input data are complex, like textual information and feature extraction is complicated.

CNN

CNN or Convolutional neural network are type of feedforward deep learning networks. CNN networks are quite easy to train and can generalize better then fully connected networks. The input of CNN is multidimensional array.

As of now, convolutional neural networks or architectures based on them are superseding or are close to supersede human abilities in many areas.

Convolutional network is usually structured into series of different types of layers:

- Convolutional layer is layer including filters that are convoluted with the input. Each filter is equivalent to a weights vector that has to be trained.
- Fully or sparsely connected layer.
- Max-pooling layer.
- Final classification layer.
- Language of context.

Results between layers are passed through a non-linearity. Nowadays, the most widely used non-linearity is rectified linear unit (ReLU):

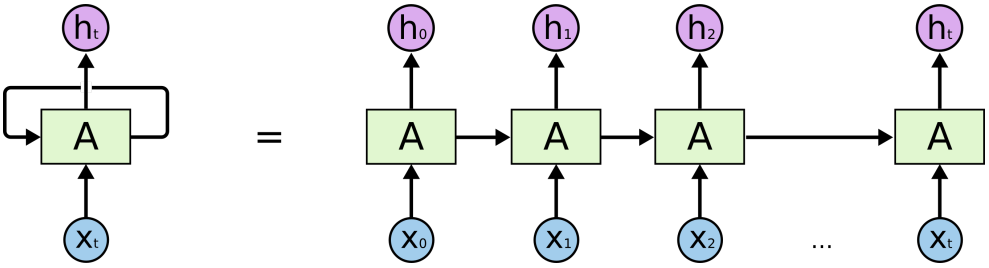

Figure 0.8: RNN schema

$$f(z) = max(z, 0) \tag{8}$$

The reasons for this architecture are that local groups of values are highly correlated and motifs can appear in any part of image or signal. Mathematically, the filtering operation is discrete convolution, hence the name.

Pooling layers are used to merge semantically similar features into one. This layer usually computes maximum of a local patch of units. Thereby, they reduce the dimensions of the representation and invariance to small shifts.

By assigning a softmax activation function, a generalization of the logistic function, on the output layer of the neural network (or a softmax component in a component-based network) for categorical target variables, the outputs can be interpreted as posterior probabilities. This is useful in classification as it gives a certainty measure on classifications.

The softmax activation function is:

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{c} e^{x_j}} \tag{9}$$

RNN

A recurrent neural network is as CNN a class of artificial neural networks. Unlike CNNs, which are feedforward neural networks, RNNs is processing sequences of inputs thanks to its internal state memory. Because of processing sequences, each input in RNN is influencing the internal state as a whole and so changing the model.

- Finite impulse RNNs is a directed acyclic graph and so has similar capabilities as CNNs
- Infinite impulse RNNs is a directed cyclic graph which makes its topological structure different that CNNs.

It have been discovered that RNN have problem with handling long-term dependencies. The problem was addressed by Hochreiter (1991) and Bengio, et al. (1994), this led to discovery of LSTM networks.

LSTM

LSTM networks or Long Short Term Memory networks is special kind of RNN. Unlike RNNs, LSTMs are capable of learning learning long-term dependencies. In last few years helped to improve many problems, where training data is sequential information, like natural language understanding tasks. LSTMs are sometimes called also Feedback Neural Network describing their behaviour.

In 2015, using LSTMs and word embeddings led to improvement of Google voice search performance by 49%.

Relation Extraction from Wikipedia Articles

This chapter specifies the task of relation extraction from Wikipedia texts. The main purpose of further specification and focusing on specific relations is to provide better and easier evaluation and in-depth insights.

Whilst focusing on specific relations, this work tries to avoid any relation specific methods, so that the whole developed relation extraction method could be easily replicated on any similar relations.

Further in this chapter is described the whole process of extraction, transforming DBpedia data, classification and tools and methods used.

Domain specification

This thesis is focused on relation extraction using classification.

Because topic of relation extraction is very large, for purposes of faster evaluation and implementation, this thesis focuses on relations from medical field as those relations are not in DBpedia database.

Selected relations: treats, prevents, causes.

The relations can be represented as combinations of types and categories:

- <medicament, drug, supplement, chemical, treatment, procedure>
- <treats, causes, prevents>
- <disease, condition, effect>

Problem definition

Problem of relation extraction can be divided into 4 categories. First by classification into Multi-class of Multi-label, then by input context into pairs of entities without context and pairs of entity mentions within context.

Multi-class classification

Multi-class or multinomial classification is problem of classifying if relation belongs to one of the specified classes. Multi-class classification does not allow for one relation to correspond to multiple classes. In relation terminology it can be described as one to one.

Multi-label classification

Multi-label classification is generalisation of Multi-class classification. Multi-label classification is problem of assigning classes to relation. In relation terminology it can be described as one to many.

Input as pair of entities without context

Often used with distant supervision, situation, when between pair of entities is known relation, but there are no labelled sentences the include the entity mentions.

Input as entity mentions with specific context

Used usually with bootstrapping, when there are labelled sentences with specific context defining the relation.

Data Analysis

DBpedia provides unstructured Wikipedia article texts in NLP interchange format (NIF) as TQL format and TTL format. The data are divided into multiple files. The main 3 files are nif-context, nif-text-links and nif-page-structure.

In this thesis the last English version of DBpedia NIF dataset is used. This dataset was released at 10 Feb 2017. Datasets used in thesis can be found in DBpedia downloads website.These datasets serve as groundwork for NLP fact extraction.

The following table contains files from main DBpedia dataset for NLP tasks. Total uncompressed size of this dataset is 384.91 GB. This dataset is not annotated, unpopulated with additional links and is not separated into sentences. Can be expected that size of fully processed dataset is going to multiply the final size.

nif-abstract-context_en.ttl and nif-context_en.ttl structure

In following lists, the structure of DBpedia NIF files is shown.

Context files provide information about the text itself as it is shown in .

file name	compressed size	total size
nif-abstract-context_en.ttl	1 GB	7.2 GB
nif-context_en.ttl	4.5 GB	19 GB
nif-text-links_en.ttl	6 GB	203 GB
nif-page-structure_en.ttl	0.5 GB	154 GB
labels_en.ttl	0.2GB	1.5 GB
category_labels_en.ttl	0.02 GB	0.21 GB
total	12.23 GB	384.91 GB

- context as unstructured string
- begin index of context
- end index of context
- source url of context
- language of context

nif-text-links.ttl structure

- type (word/phrase)
- reference context
- begin index of link in context
- end index of link in context
- anchor string value of link
- URI of link

nif-page-structure_en.ttl structure

Page structure represent how the context of wiki page is structured. How the context string is separated into paragraphs, and sections.

- type (paragraph/section)
- reference context
- begin index of paragraph or section
- end index of paragraph or section

RDF

RDF or Resource Description Framework is W3C standard. It was made to represent resources on the web and relations between them. The information in RDF is represented by <subject><predicate><object> called triple.

RDF uses URIs to name relationship between resources and to define triples. Set of triples represents a graph like structure, where URIs resources represent nodes in the graph.

RDF can be described in various notations:

- TTL – Terse RDF Triple Language
- TQL
- RDF/XML – XML format
- N-Triples
- JSON-LD

SPARQL

SPARQL Protocol and RDF Query Language are created by W3C. SPARQL is RDF based query language, it enables to manipulate and retrieve data from DRF based database. SPARQL, together with RDF is one of key technologies for semantic web or Web 2.0.

SPARQL allows a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns.

Example of SPARQL query describing selection of all distinct properties related to Person based on rdf schema:

```
select distinct ?property where {
  ?property
  <http://www.w3.org/2000/01/rdf-schema#domain>
  <http://dbpedia.org/ontology/Person>
  . }
```

NIF

Natural Language Processing Interchange Format is RDF/OWL-based format that aims to achieve interoperability between Natural Language Processing (NLP) tools, language resources and annotations[18].

All NIF ontology classes are derived from the main class nif:String. nif:String represents simple string of Unicode characters and each URI is nif:String subclass.

According to DBpedia there is 6,078 Disease entities in English version

Reading and parsing DBpedia NIF dataset

For reading NIF dataset, variety of tools can be used, but file sizes are to big to use these tools effectively on non-server based machine.

Creating subdataset based on targeted domain

For speeding up the process of working with the dataset, and so speeding up the process of testing and development, the DBpedia dataset was filtered to create subdataset based on the targeted domain of medicine and relations: treats, causes and prevents.

Filtering method description:

1. By analysing DBpedia entities related to the domain, set of DBpedia properties describing selected domain was created.
2. These properties were used to query all corresponding URIs.
3. Based on the list of related URIs all pages involving any of the URIs were selected.
4. Resulting set of pages was used to filter the DBpedia dataset.
5. Filtered DBpedia datasets were created.

Example of DBpedia entity types for domain selection.

```
'dbc:Chemical_elements',
'dbo:ChemicalSubstance',
'dbc:Toxicology',
'dbo:ChemicalCompound',
'yago:Drug103247620',
'yago:Analgesic102707683',
'yago:Anti-inflammatory102721538',
'yago:WikicatDrugs',
'yago:WikicatAnalgesics',
'yago:WikicatDrugs',
'dbo:Drug',
'yago:Therapy100661091',
'yago:Inflammation114336539',
'yago:Illness114061805',
'yago:Ailment114055408',
'dbo:Disease',
'dbc:Pain',
...
```

Table 0.1: Filtered datasets of targeting domain

filtered file name	**total size**
nif-abstract-context_en.ttl	0.1 GB
nif-context_en.ttl	0.31 GB
nif-page-structure_en.ttl	2.5 GB
nif-text-links_en.ttl	3.2 GB
labels_en.ttl	0.1 GB
total	6.21 GB

These Filtered DBpedia datasets can be found at the included storage.

Total number of wikipedia pages contexts included in this filtered dataset is 58455.

Data prepossessing

The information about context, links and structure is divided into 3 TTL files, representing RDF structure. To get a better understanding and to populate context with links, the information about DBpedia resource, corresponding links, structure and context are restructured into objects representable in JSON format.

Link

Link object represents annotation – link to DBpedia resource has following structure:

- LinkID – URI+LinkType+start+end,
- URI – URI resource link is reffering to,
- start – start index of link in context string,
- end – end index of link in context string,
- label – label of the link is referring to,
- surfaceForm – the surface form of the link as word or phrase representation
- probability – probability of link representation corresponds to correct URI,
- synonyms – list of synonyms.

Probability property of the object is not required value, for links extracted from the DBpedia dataset is probability considered always as 100%.

Page

Page object represents information about the DBpedia resource context and corresponding links:

- URI – URI resource used also as ID,
- start – start index of context string,
- end – end index of context string,
- label – label the resource is referring to,
- sourceUrl – wikipedia URL,
- text – the context string,
- links – list of Link objects.

Enriching the dataset with additional links

In Wikipedia, anotators are required to link corresponding entities only with their first mention. That is the reason DBpedia dataset ia annotated only by those interlinked entities. The most of the interesting information in dataset is still not linked to corresponding resource. For that reason dataset needs to be enriched by additional links.There are various tools in field of Entity linking and recognition.

For purposes of this thesis we define terms we will use to describe these tools:

Named entity

Named entity is real-word named object such as person, location or organisation, that can be expressed by word or phrase.

Examples:

- Barack Obama – person,
- New York – city,
- Google – organisation,
- Wednesday – date.

Entity

Entity is any real-word concept that can be described by word or phrase.

- Barack Obama – person,
- fear – feeling,
- New York – city,
- trash – object,
- Google – organisation,
- market – place,
- drug – thing,
- Wednesday – date.

Linking

Named-entity linking, or entity linking is a sub-task of information extraction that links an entity in unstructured text to an existing corresponding resource, usually knowledge database. For linking such entity unique identifier is used, usually an URL address.

Recognition

Named-entity recognition or entity recognition subtask of information extraction that classify entity in unstructured text into predefined categories such as person names, organisations, locations, medical codes, time expressions, percentages. Also known as entity identification, entity extraction or entity chunking.

NER and ER tools

Because NER is solving problem of population of dataset with links only partially, in this thesis no in-depth analysis of NER is described. Only list some of the state of art tools and methods, that could be used furthermore for task of classification.

Most used NER taggers:

- spaCy NER Model,
- Stanford Named Entity Recognizer.

Table 0.2: Comparison of NEL and EL tools on different datasets and their average performance. with F1 macroscore and F1 microscore

F1@MA F1@MI	AIDA A F1@MA	AIDA B F1@MA	DBpedia Spotlight F1@MA	AIDA A F1@MI	AIDA B F1@MI	DBpedia Spotlight F1@MI
FREME	23,6	23,8	37,9	37,6	36,3	52,5
FOX	54,7	58,1	11,7	58	57	15,9
Babelfy	41,2	42,4	51	47,2	48,5	51,8
Entityclassifier.eu	43	42,9	19,9	44,7	45	25,5
DBpedia Spotlight	49,9	52	70,1	55,2	57,8	72,6
AIDA	68,8	71,9	21,4	72,4	72,8	24,9
WAT	69,2	70,8	8,3	72,8	73	67,1
Wikifier	NA	NA	NA	NA	NA	NA
end2end_neural_el	86,6	82,6	80,5	89,4	82,4	80,1

NEL and EL tools

Because NEL and EL tools and methods will directly affect training dataset and its precision, in-depth analysis of existing tools and methods is necessary. If EL tool is using Wikipedia or DBpedia URIs for linking, this tool is often called Wikifier and the process called Wikification.

Benchmarking tools for NEL and EL

Because NER and NEL are one base problems of NLP, there has been few attempts to compare them.

GERBIL is general entity annotation system used for benchmarking different NEL and NER tools on multiple datasets, allowing for direct comparison of these tools[19]. Unfortunately not all tools was and could be compared. To addition GERBIL tool has some problems updating results to some of the tools.

To complete comparison website Paperswithcode.com and similar resources were used. Paperswithcode is website allowing for comparison of different tools and algorithms for certain tasks using specified datasets. Nlpprogress.com is website focusing on tracking development in NLP tasks[20] nlpcomparer.

NEL and EL tools comparison

Using those resources and websites of found Tools data in tables 3 and 4 were collected, allowing for comparison most interesting NEL and EL tools.

In table 0.3 is list of compared most successful tools for NEL on 3 most popular datasets and their availability.

Table 0.3: Comparsion of tools and their availability with links to demos.

F1@MA F1@MI	Available as a service	Code available	NEL/EL	Demo	Source
FOX	Y	Y	NEL	demo	github
Babelfy	Y	N	EL	demo	
Entityclassifier.eu	Y	Y	NEL/EL	demo	github
DBpedia Spotlight	Y	Y	EL	demo	github
AIDA	Y	Y	NEL	demo	github
WAT	N	N	NEL		
Wikifier	Y	N	EL	demo	
end2end_neural_el	N	Y	NEL		github

Based on availability, accuracy, possibility to link entity mentions, not only named entity mentions, speed and performance on DBpedia dataset and performance on example from extracted dataset. DBpedia Spotlight was chosen as main annotator.

DBpedia Spotlight

DBpedia Spotlight uses four step analysis on given text, performing named entity extraction, entity detection and name resolution, so also disambiguation[21]. DBpedia Spotlight does not focus only on Named entity extraction but also on Entity extraction and annotation in a broader sense, which is important for purposes of this thesis.

DBpedia spotlight allows for solving multiple related tasks:

- spotting,
- annotation,
- disambiguation,
- candidates selection,
- entity filtering.

Spotlight can be used in several ways, as a Web Service through API, as a jar with all dependencies included, building project form the source code, or thought maven, using Scala plugin to run classes from command line or as a docker image.

Implementation

Simple python API client was build for purpose of annotating the dataset. Because text for purposes of annotating can be tens of thousands characters

long, the text needs to be split into shorter strings, send to DBpedia spotlight and resulting annotations then fixed and reindexed passed on the string partitioning.

All transforming functions can be found in transformers directory.

However, annotating this way the 0.3 GB of text through DBpedia Spotlight API would take several days.

For purposes of shortening this amount of time, dataset was split for purpose of parallel processing into several parts.

Several DBpedia Spotlight instances were deployed on Google Cloud VM which annotated texts in parallel.

Annotated texts were then merged and transformed using transforming function into JSON lines file of in Page format described in previous section.

Deployment

As a Deployment method of DBpedia Spotlight the deployment of Docker Image is used.

Several Google cloud VMs were configured and deployed with docker.

Then docker image is pulled from Docker Hub and run.

```
docker pull dbpedia/spotlight-english
docker run -i -p 8080:80 b8d96addc33d spotlight.sh
```

Coreference resolution and linking

Large amount of entity mentions in text is represented as a reference, usually as pronoun. The importance of resolving these references are even more important, because of the nature in which they appear. Entity is often being described in one sentence, effects and other relations to other entities is being described in other sentence using reference.

Examples:

- Historically, it has been used to treat nasal congestion and depression.
- Specific inflammatory conditions in which it is used include Kawasaki disease, pericarditis, and rheumatic fever.

Unfortunately none of the analysed EL tools are taking coreference resolution and linking into a count.

Definitions

Definition 0.0.6. Coreference is when two or more expressions in a text refer to the same entity. Expressions having same refferent[22].

Definition 0.0.7. Coreference resolution is the task of finding all expressions that refer to the same entity in a text[23].

Definition 0.0.8. Anaphora is the use of an expression whose interpretation depends upon expression in context.

Definition 0.0.9. Anaphora resolution is process of determining the antecedent of the anaphora.

In context of DBpedia Pages, where context of sentence is often context of the page as a whole, the coreference and anaphora resolution has special conditions.

Analysis

Analysing over hundred pronoun occurrences in DBpedia dataset following observations were made.

Where gender of pronoun corresponds to gender of context entity

- 91% of pronouns appearing in abstract is referring to the Page entity context, if corresponding genders,
- 87% of pronouns appearing in the text refers to the entity, if corresponding genders.

Combining with simple regular expression, to filter out phrases with general *it*:

- it has been shown that,
- it has been seen that.
- it is known that.

Filtering out these phrases allows for precision of near 95% reached.
However for other 2 genders, additional algorithm is needed.

Anaphora resolution tools

Because not all anaphoras correspond to the context of the page itself, but correspond to entity mentioned in sentence before, research about anaphora resolution was done. From the found tools neuralcoref as it is integratable to python using pip and it is the state-of-the-art coreference resolution system.

Analysed tools:

- Stanford CoreNLP
- neuralcoref

- Hobbs algorithm
- RAP algorithm

Algorithm

The final algorithm for coreference resolution using observation and neuralcoref tool was developed. Pseudocode of the algorithm can be found here.

```
tokens=tokenize(sentence)
corefs=processCoref(sentence)
for token in tokens:
    if token not in phrasesExeption:
        if inAbstract and isPronoun(token) and matchGramGender(token):
            addLink(pageLink)
        else:
            if token in corefs
                if getCoref(token) in linkLabels:
                    addLink(link)
                elif getCoref(token) is pageLabel:
                    addLink(pageLink)
            elif matchGramGender(token)
                addLink(link)
```

Preparation of the dataset

After enrichment with additional links and coreference resolution, the dataset of annotated Pages is transformed into dataset of sentences. Structure of the sentence object is similar to the structure of Page object.

Sentence splitting

For sentence splitting of text, nltk sentence tokenizer is used. The analysis of the outputs has shown that nltk is not good in splitting text with small errors and misspells.

Found sentence splitting errors:

like no space after end of sentence, or chemical formulas. Sentence splitter was updates with additional rules:

Based on these foundings, these errors were eliminated by adding additional rules to the algorithm.

After sentence splitting is complete, the annotated dataset is analysed. Total number of splitted sentences in dataset is 2 256 329, total number of sentences with more then 2 medical related entity is 717 685. Because our problem specification is defined as labeling entity pairs, sentences with one or less annotated entity mentions cannot be used for further relation extraction.

- No splitting of sentence, where no space after „.“ is found.
- Problem with splitting Chemical formulas.

Figure 0.9: Types of sentence splitting errors

Table 0.4: Comparation of sentence splitters

algorithm	**errors** %
nltk.tokenizer	2.3%
with additional rules	0.04%

Part-Of-Speech Tagging

Part-Of-Speech Tagging allows for adding more information to the words in sentence itself.

As Part-Of-Speech Tagging NLTK POS tagger was used. It is still considered as state-of-the-art. NLTK POS tagger is now using machine learning methods to classify the tokens.

Entity categorization

Thanks to entity linking, the information about entities from DBpedia can be used to categorize entity mentions in the text.

For example: The mutation [effects] that causes the autism [diseases] is not present in the parental genome. In this sentence, entity mutation has DBpedia properties which can be mapped to one or multiple groups, that has been created for this thesis.

Dataset statistics

Most entities that appeared together

Clearly the reason for Cancer and type of cancer to appear together is that one is hypernym of the other.

- 2873 apearences of Cancer – Colorectal cancer
- 2619 apearences of Cancer – Cervical cancer
- 2607 apearences of Cancer – Thyroid cancer
- 2567 apearences of Cancer – Esophageal cancer
- 2549 apearences of Cancer – Gastrointestinal cancer

Table 0.5: List of some of the groups based on DBpedia properties

URI groups	**in URI identifiers belonging to this group**
diseases	13270
chemical_substances	2019
effects	14728
analgesic	207
anti_inflamation	110
antibiotics	319
bacterial_diseases	138
viral_diseases	683
antivirotics	108
pain	1
fever	1
inflamation	1

Most appeared groups together in one sentence.

times of appearance	**group pair**
1235779	effects effects
1177796	effects diseases
1134350	diseases diseases
32834	chemical substances effects
30696	chemical substances diseases
9061	viral diseases effects
8966	pain effects

Stop words removal

Some types of text, like social media texts, is full of noise, words that does not hold meaning. Fortunately, Wikipedia does not belong to this group and so effect of stop words removal will not have such strong effect. However the effects of stop words removal is further explored in the experimentation chapter.

Stemming and lemmatization

Text Stemming is described as modifying a word using multiple linguistic processes to find the stem. For example, the stem of the word ”learning” is ”learn”.

Lemmatization refers to elimination redundant prefixes and suffixes of a word to get the base word (lemma).

Because Lemmatization removes big amount of information, the lemmatization will be part of the experimentation process and will not proces the dataset istelf.

POS Tagging

POS Tagging or Part-of-speech tagging or word-category disambiguation is process of tagging tokens in a sentence with part of speech category tag. POS Tagging allows for improving the number of information text holds. Its importance will be part of experiments.

Proposed relation extraction method

In this thesis we propose relation extraction method combining bootstrapping technique and distant supervision.

- Using Bootstrapping and distant supervision knowledge from DBpedia, we create small set of triples for each relation
- For each set of found triples, the corresponding context, for those triples will be extracted
- Context with entity mentions will be used to train a classifier and create model
- Created model will be used to classify relations in annotated dataset and rate with probability score
- Classified relations between entity mentions and their probability will be used in probability combiner to get high quality set of fact triples.

Creation of Training datasets

Because the goal of this thesis is to explore possibilities without existing hand-labeled datasets we will rely on bootstrapping, distant supervision and their combination for creation of training data.

For purposes of training data creation function getTrainingDataset was implemented, which combines Bootstrapping algorithms and Distant Supervision based on given parameters.

Bootstrapping

Based on how bootstrapping is used, 2 types of data can be created based on sentence context.

- Finding entity pairs with existing relation

- Finding entity mentions with sentence context

Finding entity pairs with existing relation

Finding entity pairs, without the sentence context can lead to Training data with huge amount of noise. This training data can be filled with wrong relations or sentences where there is no relation between entity mentions.

This way even when bootstrapping is used, the training data are similar to distant supervision training data.

Finding entity mentions with sentence context

If entity mentions with sentence context is used as Training data, the bias created by bootstrapping is going to be learned by the model. The trained model will not be able to easily generalise.

Distant Supervision

Because the relations we explore in this thesis are defined, we can use DBpedia properties to define subsets of entities, where certain relation is defined.

For example set of entities with property type: viral_disease is treated by entities with type: antivirotics.

Of course that doesn't mean that each pair of viral disease and antivirotic has relation to treat.

Also, that doesn't mean every sentence mention of viral disease and antivirotic has relation to treat.

Algorithm

The algorithm is using similar technique as DIPRE and Snowball algorithm and improving upon them by adding heuristic and statistical filtration and additional information provided by DBpedia properties thanks to entity linking.

The found entity pairs are rated with probability score based on sentence similarity to seed sentences and already found sentences and entity distances.

Algorithm parameters are

- list of pair entity mentions with context sentence,
- set of regular expression rules,
- set of entity URIs corresponding to the first entity,
- set of entity URIs corresponding to the second entity,
- number of cycles,

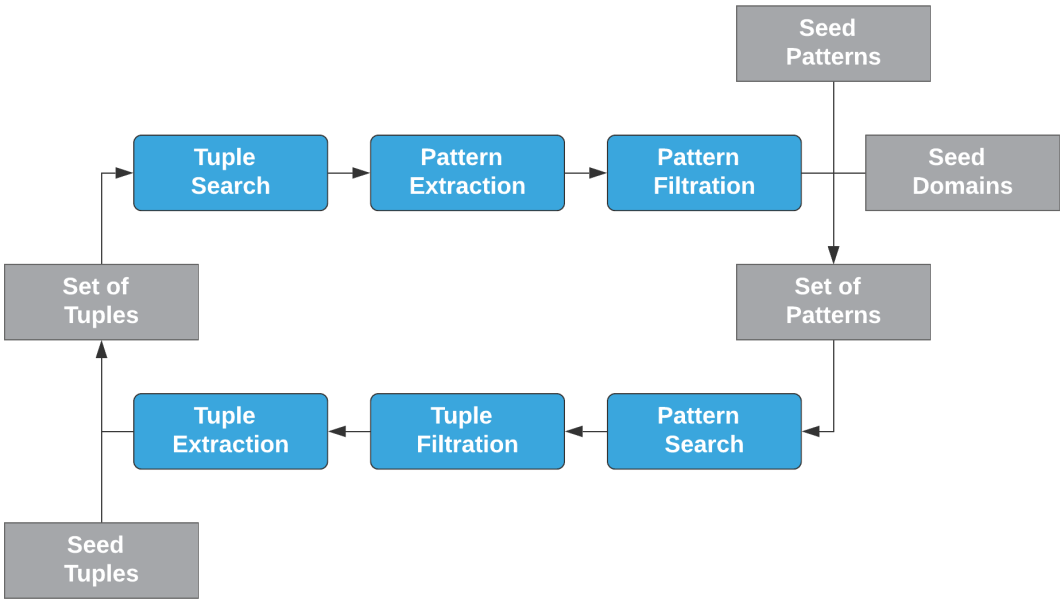

Figure 0.10: Bootstrapping process

- probability cut Off,
- type of result – with sentence context/without sentence context,
- isCutOff at last iteration.

Training dataset creation

We used our bootstrapping method to find several facts. The structure of this dataset is simple, it is set of triples <relation> <entity1> <entity2>. To create training dataset we find these entity pair mentions in the annotated DBpedia dataset.

The hope is to find more relevant context representing the relation. There is also big probability, that some of the text context found around entity mentions will not represent labeled and searched relation. To make the errors small as possible, the final training dataset will consist of 3 parts.

- instances of sentences with entity mentions representing the relation
- instances of sentences with entity mentions for which it is not possible to represent the relation
- random mentions labeled as not representing the relation

Table 0.6: List of positive datasets, corresponding to the relation

relation	size-type	#sentence mentions	# entity pairs	estimated accuracy
treats	small	28242	421	96%
treats	medium	83204	796	82%
treats	large	96424	1673	78%
prevents	small	14262	145	94%
prevents	medium	25538	534	78%
prevents	large	59582	1245	72%
causes	small	20760	347	86%
causes	medium	28648	827	76%
causes	large	31264	1378	74%

Model development

In this section we describe technologies used to create models. For purpose of comparison and finding the best model, several models will be created.

TensorFlow

TensorFlow[24] is open-source library focused on expressing machine learning algorithms and an implementation for executing them. It is developed and maintained by Google. It can be used in python and is used for machine learning applications like neural networks and deep neural networks.

However TensorFlow is very detailed library for developing neural networks. It is not as user-friendly as frameworks build upon it. Because of the focus on methodology, training data creation and my experience with neural networks as software engineer student, TensorFlow will be used through frameworks build upon it. The two most used frameworks are Keras and PyTorch.

Keras

Keras[25] is an open source high-level neural networks API. It abstracts over multiple machine learning libraries such as TensorFlow, Microsoft Cognitive Toolkit, R or Theano. It is designed for fast experimentation with deep neural networks. Its website provides a good documentation and its code is open source under MIT license on GitHub.

Recently Keras has been also directly integrated into TensorFlow package and can be accessed through tf.keras.

PyTorch

PyTorch [26] is an open source machine learning library. It is based on Torch and used for applications like natural language processing. PyTorch was developed by developed by Facebook's AI research group.

Ktrain

Ktrain[27] is Keras wrapper. It is a library which makes it easier to configure, test and deploy Keras models. Ktrain specializes on Neural networks and deep learning type of networks.

Word embedding

Word embedding is name for technique of mapping words or phrases to vectors of real numbers.

Vectors are being generated by one of many vectorization methods including neural networks, reduced word co-occurence matrix or probability models.

One of limitations of word embedding is that word embeddings does not count for sentence context, does not count for polysemy and homonyms with exception of BERT and ELMo.

List of popular word embedding tools:

- weighted words,
- TF-IDF,
- Word2vec,
- GloVe,
- FastText,
- ELMo,
- BERT.

Word2vec

Word2vec [**?**]is a tool that allows for implementation of CBOW and skip-gram architectures. This architecture allows to learn relationship of word to its text context as a vector giving vector representation of the word, translating word to vector. These vector representations had great impact in many NLP applications and problems.

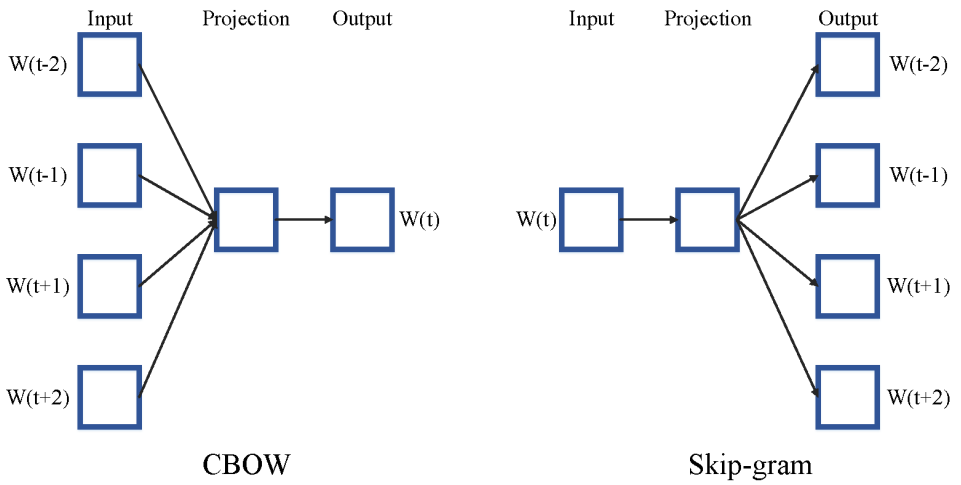

Figure 0.11: Continuous Bag of Words Model (CBOW) and Skip-gram model for word representation learning

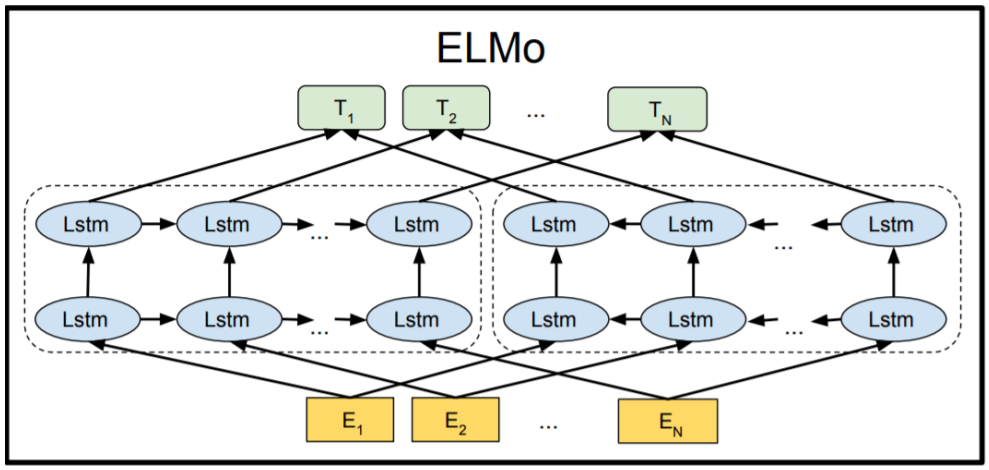

Figure 0.12: ELMo architecture using Lstm network

ELMo

ELMo is a deep contextualized word vectorizer. Compared to traditional word embedings, ELMo takes whole sentence or paragraph and returns word vector representations taking sentence context in account, resolving polysemy and homonyms. These word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus[28].

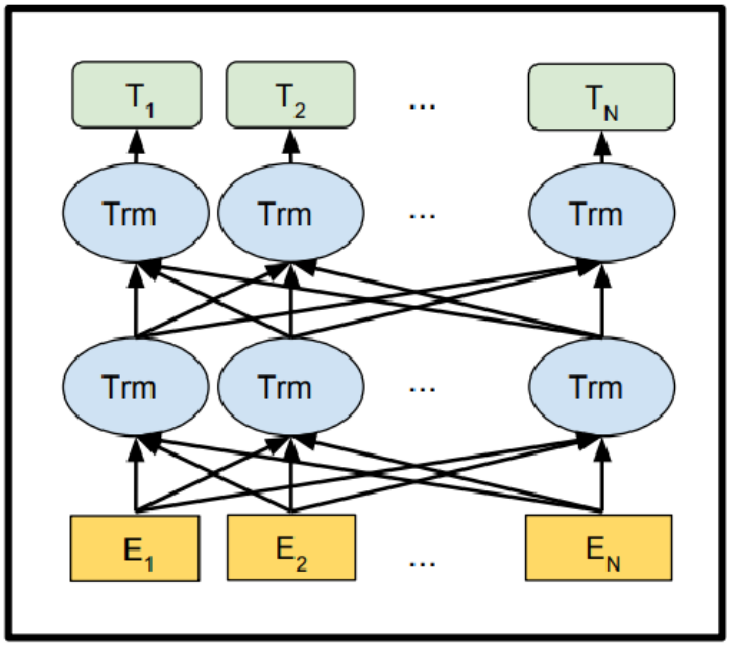

Figure 0.13: BERT architecture

BERT

BERT (Bidirectional Encoder Representations from Transformers)[29] is in a sense an ancestor of ELMo. BERT however uses transformer architecture0.13 to compute word embeddings. It has been shown to produce excellent word embeddings, achieving state-of-the-art results on various NLP tasks in 2019.

Bag of words

Bag of words is method for transforming text into processable feature vector. Bag of words vector represents words in text as vector of word counts, where each index of vector represents number of word mentions in vectorized text.

Example Sentences

- <drug> treats <disease>
- <drug> is useless for <disease> treats
- Article treats <drug> as <drug>.

Table 0.7: Sentences represented as vectors using Bag of Words

Article	<drug>	treats	<disease>	is	useless	for	treats	as
0	1	1	1	0	0	0	0	0
0	1	1	1	1	1	1	0	0
1	2	1	0	0	0	0	1	1

Proposed models

Logistic regression model

As our baseline model we use Logistic regression and Bag of Words with features. Second possibility is to use Tf–idf term weighting instead Bag or Words. To Create Logistic Regression model library Scikitlearn is used.

BERT and LSTM model

At last new technique, combination of BERT word embedding with LSTM. In this approach we use multiple hidden layers and fine tuned BERT model.

Experiments

In this chapter we experiment using different training datasets, features tools and models. The goal of this experimentation is to find the best classification model for our task.

Virtual Machine

For Experiments and main task of relation extraction, especially because of intended use of deep learning, there is need for powerful machine with even more powerful GPU.

Parameters of the computer used for experiments and relation extraction shown in Table 0.8.

Evaluation metrics

Because we extract totally new relations, we do not work with hand-labeled and corrected datasets, evaluation can be done by 3 ways. Evaluation by bootstrapped dataset, approximation by manual evaluation of the classification output, specialized sample dataset.

Table 0.8: Testing Virtual Machine parameters

Part	Description
CPU	2vCPU @ 2.2GHz
MEM	13GB RAM
GPU	Tesla K80 GPU 350 GB
OS	Linux

Approximation by manual evaluation

To approximate precision is not that difficult, as in our relation detection problem there is fare more positively labeled instances then negative labeled instances.

Problem is with validation of falsely labeled positive instances. Based on dataset analysis, there is less then 1% of entity pair mentions corresponding to the relation. The other more then 99% has different relation or no relation at all defined.

$$\hat{P}recision = \frac{\#of - correctly - extracted - relations - in - the - sample}{Total\#of_extracted_relations_in_the_sample} \tag{10}$$

To approximate recall is much more difficult task, as it is hard to estimate what is the total number of positive instances. We can estimate the percentage of true positives and false negatives in analyzed sample and use these estimates to calculate Recall. Recall defined this way is very much inaccurate and so will be calculated only in special instances.

$$\hat{R}ecall = \frac{\%ofcorrectlyextractedrelationsinthesample}{\%ofrelevantrelationinthesample} \tag{11}$$

Accuracy has similar problem as Recall and so will be mentioned only in special instances.

$$\hat{A}ccuracy = \frac{\#oftruepositiveinthesample + oftruenegativeinthesample}{Totalsamples} \tag{12}$$

Because F1 score depends on Recall, is also highly inaccurate and can be only approximated.

$$F1 = 2 * \frac{Precision\Delta Recall}{Precision + Recall} \tag{13}$$

Evaluation partition of training dataset

The created training dataset can be split into training and test dataset and then validated by precision, recall, and F-score. However, this training dataset can be biased and is also, the dataset is not 100% accurate. This way the classifier can learn pattern that does work only for training data.

Specialized testing dataset

Specialized testing dataset is hand labeled dataset with special properties. Because it is really hard to find accurate labeled datasets representing real

Table 0.9: different training dataset comparison

Dataset	positive %	negative %	Precision	Recall	F1 score
small	59%	41%	0,053	0,917	0,100
medium	38%	62%	0,692	0,087	0,154
large	18%	82%	0,807	0,846	0,826
huge	11%	89%	0,779	0,899	0,835

data for purposes of this thesis, 3 specialized datasets were made. These datasets does not represent the classification

Each of these datasets were in following way:

- dataset was filtered by keywords [treat, cause, prevent] to get sentences with 2 entity mentions for corresponding relation classification
- from these found sentences 100 positive and 100 negative examples was found

These specialized datasets validate if classification can recognize relation a between which entity mentions this relation belongs.

Logistic Regression experiments

As our models are binary classifiers, each of the model classifies whether instance has or doesn't have relation. Result of the model as a whole is set of labels describing classified instance.

For example <entity1> or antibacterials are a type of antimicrobial used in the treatment and prevention of <entity2> will be labeled as treats and as ṕrevents:

As first set of experiments we use small training datasets with high accuracy. These datasets and their creation was mentioned in previous chapter. To train the model, vectorization has to be applied.

As testing datasets in these experiments, the manually created datasets were used. These datasets involve 100 examples of instances representing targeted relation and 100 examples, that does not represent targeted relation.

Different training datasets comparison

In this experiment we use 4 different training datasets. All used datasets in this experiment are derived from small positive dataset for relation treats.

From this positive dataset 4 training datasets are created and validated against testing data. Each of this 4 datasets differs in number of negative instances and so differs in positive/negative labels ratio.

Table 0.10: model validated by testing dataset

Relation	Precision	Recall	F1 score
treats	0,871	0,703	0,778
causes	0,810	0,772	0,790
prevents	0,793	0,724	0,756

Table 0.11: model trained by only middles

Relation	Precision	Recall	F1 score
treats	0,850	0,817	0,833
causes	0,737	0,785	0,760
prevents	0,805	0,851	0,827

As we can see in Table 0.9 Dataset with best scores are large and huge datasets, but because Precision is much more important for information extraction, large dataset with positive/negative ratio around 20:80 will be used.

BOW unprocessed

In this experiment we vectorize whole sentences with entity mentions and evaluate Regression . For vectorization of the pair of entity mentions and their context, BOW representation is used. Text will be converted into BOW representation using CountVectorizer from sci-kit learn library. For this experiment no additional features will be added. Results of this experiment are shown in Table 0.10.

The results of 0.10 are quite good. This experiments proves that not only classifier can recognize most important keywords [treat, prevent], but also if found relation corresponds to the entity mentions.

Using only middles of sentences

As second experiment we try to train the model only by context appearing between entity mentions. Because context appearing between entity mentions does usually define the relation. Results of this experiment are in Table 0.11. As we can compare scores with scores from previous table we can observe that relation treats and prevents benefit from this feature, but relation causes does not. That might be because of how the word cause is used. Maybe this relation is defined less often in middle of the sentence.

POS tagging

As our third experiment we add feature of POS tagging 0.12.

Table 0.12: model trained by only middles with POS tagging

Relation	Precision	Recall	F1 score
treats	0,804	0,850	0,827
causes	0,789	0,898	0,840
prevents	0,749	0,868	0,804

Table 0.13: model trained by only middles with Stemming and Lemmatization

Relation	Precision	Recall	F1 score
treats	0,859	0,558	0,676
causes	0,717	0,803	0,757
prevents	0,746	0,677	0,709

Table 0.14: Sensitivity and specificity for relation treats

Dataset	True	False
Positive	22	72
Negative	99	1

Stemming and Lemmatization

This experiment we explore the effect of Stemming and Lemmatization 0.13.

Evaluation of main annotated dataset relation classification

In this experiment, we use preceding model to predict values on sample from annotated dataset. This experiment is evaluation of the relation extraction results. Sample of predicted found relations and predicted no relation was Analyzed. As we can see in Table **??** the accuracy of found relations, the precision is quite low. That indicates that the testing data and training model does not define scope of the problem correctly. However recall is quite high. Unfortunately precision is most important parameter for reliable fact extraction.

Summary

Experiments used on Logistic Regression and BOW proved, that model trained on selected trained data is biased and does not generalize well. Created model has great capabilities to classify correct relation in sentence, where some of the

Table 0.15: Sensitivity and specificity for relation causes

Dataset	True	False
Positive	18	87
Negative	94	4

Table 0.16: Sensitivity and specificity for relation prevents

Dataset	True	False
Positive	12	64
Negative	100	0

key keywords are mentioned, but cannot recognise more complicated sentence meanings.

Example of wrongly classified positive relations.

- Like all <entity1>, it is not useful for the treatment of <entity1>.
- While daily <entity1> can help treat a clot-related stroke, it may increase risk of a <entity1>.

Deep learning experiments

In this section we experiment with the model based on BERT word embeddings and LSTM artificial neural network.

When experimenting with neural networks, before training of the neural network begins hyperparameters can be modified to achieve the best results.

Because Neural networks require static input length, training data has to be shortened or appended by padding.

Main hyperparameters of our model are

- max length of instance input
- max number of embedding features
- number of epochs
- batch size
- number of iterations
- learning rate

BERT is rather memory-intensive to work with.

Because of number instances our neural network is trained on, it can be tens of thousands up to hundred thousands, the batch size has to be small. Max number of embedding features has also impact on memory and so it is better not raising it too much. Number of epochs for standard text classification tasks is 1-6. we Depending on max number of embedding features will allow for more or less instances

For purposes of model training the Ktrain framework is used. Ktrain framework allows us to aproximate optimal learning rate automatically.

	precision	recall	f1-score
class1	0.51	0.98	0.67
class2	0.88	0.13	0.23
accuracy			0.54
	0,899	0,835	

Table 0.17: loss: 0.1872 - acc: 0.9369

	precision	recall	f1-score
class1	0.86	0.87	0.86
class2	0.88	0.87	0.87
accuracy			0.87

We will use specialized Ktrain model focused on text classification.

We will train on the same datasest as the regression model, only this time without preprocessing.

Experiments using 1 Epoch

- batch size: 2
- test data of 40 000 instances
- 1 epoch
- learning rate 2e-05

Experiments faster learning rate

- batch size: 2
- test data of 40 000 instances
- 1 epoch
- learning rate 3e-05

Evaluation of main annotated dataset relation classification

Similarly as in Logistic Regression experiments, we use trained model to predict values of our DBpedia annotated dataset. as can be seen in table 0.18, results of this deep learning model using BERT are much better then our previous attempt with Logistic Regression. Because BERT is creating word embeddings and is not using only information provided in train dataset, but also pre-trained English word embedding model, this deep learning model can generalize much better.

Table 0.18: Sensitivity and specificity of BERT and LSTM model

Dataset	True	False
Positive	82	24
Negative	98	1

Fact extraction

In last chapter we implemented experimented on and evaluated trained models. Now we use the best model for fact extraction. The best model we created is deep learning model base on BERT embedding and LSTM artificial neural network. The precision of the model has been evaluated close to 80%.

Fact extraction

In this section we describe method we use for fact extraction based on previous work of trained models and dataset preparation.

1. We split dataset into smaller parts
2. We load saved models
3. Then we use build in Ktrain function for class prediction and predict classes for each instance in dataset
4. All labeled instances are then saved
5. After predicting all dataset parts, the result is merged

This way we were able to extract all pairs of entity mentions corresponding to relation including context of the mentions.

```
predictor = ktrain.get_predictor(learner.model, preproc)
predictor.get_classes()
predictor.predict(dataset)
```

Probability combiner

The main goal of this thesis is to create set of triple facts. To do that, probability combiner function was used to combine probabilities of entity mentions. This way the probability of relation mention in text context is turned into probability of relation existing itself.

Quality analysis

To get perspective on quality of the results, random sample of 100 instances was analysed. From 100 instances 89 was considered to be truthy. However to analyze this resulting data properly is difficult. In example below shows part of extracted facts without sentence context. In this example Evidence-based_medicine is said to be treating Chronic_pain. This triple can be recognized as false, but some could still agree the relation is correct.

Example of extracted facts

```
0.9989066480635329      treats
<http://dbpedia.org/resource/Paracetamol>
<http://dbpedia.org/resource/Fever>
0.9994568064395509      treats
<http://dbpedia.org/resource/Drug>
<http://dbpedia.org/resource/Fever>
0.7766895260648318      treats
<http://dbpedia.org/resource/Analgesic>
<http://dbpedia.org/resource/Ethanol>
0.9996723771964008      treats
<http://dbpedia.org/resource/Evidence-based_medicine> <http://dbpedia.org/
0.9713064840283407      treats
<http://dbpedia.org/resource/Evidence-based_medicine> <http://dbpedia.org/
0.9998187161119909      treats
<http://dbpedia.org/resource/Gabapentin> <http://dbpedia.org/resource/Neu
0.9999757757047596      treats
<http://dbpedia.org/resource/Ibuprofen>
<http://dbpedia.org/resource/Pain>
0.9826831768541949      treats
<http://dbpedia.org/resource/Topical_medication>
<http://dbpedia.org/resource/Pain>
0.9023156906346816      treats
<http://dbpedia.org/resource/Diclofenac>
<http://dbpedia.org/resource/Pain>
```

Resulting extracted set has 123086 instances of facts. 65644 of relation treats. 23624 of relation causes and 33818 of relation prevents. This extracted facts are submited to be added to DBpedia knowledge graph. This way DBpedia will be extended by completely new type of relation.

Conclusion

The goal of this master thesis was to research existing methods of fact extraction. To get familiar with the knowledge databases, implement selected fact extraction method or methods and use these methods on Wikipedia article texts. In the end extract selected fact triples from text and evaluate.

In the introduction of this mater thesis was explained relation extraction. Relation extraction was divided into multiple categories categories were described and main category of multi-label relation extraction was chosen. For relation extraction advanced machine learning techniques were researched and described. In this thesis we focused on range of relations in selected filed of medicine. This field was chosen because DBpedia doesn't have range of relations from this field of interest. After definition of the problem, the DBpedia dataset was parsed and preprocessed. For preprocessing was used several techniques. The main preprocessing of the dataset was entity linking. After dataset was enriched with additional links also Coreference resolution problem and Anaphora resolution problem was addressed.

Enriched and annotated dataset was analysed and its quality was improved over multiple iterations.

In this thesis we propose new method of relation extraction based on bootstrapping combined with distance supervision methods. This method involves automatic creation of training datasets for supervised deep learning models. Semi-automatic creation of training dataset is one of highlights of this thesis. These semi-automatically created training datasets were then successfully used to train modern deep neural network model Logistic Regression. Accuracy of these models were evaluated and best, most accurate model was used to extract facts from the annotated DBpedia dataset. These extracted facts were then submited to the DBpedia database. and method was implemented a tested for quality.

From the provided experiments and extracted data analysis we can conclude, that our proposed solution for relation extraction, subtask of fact extraction was succesful.

Future work

In the future, the pipeline for fact extraction can be further upgraded. The future of fact extraction lays in automation. Code of this thesis could be extended, so that is more automated. This thesis can be used also as baseline for automatic graph databasse extension system. In the future this work will be used for more types of fact extraction and DBpedia enriching.

Bibliography

[1] Nickel, M.; Murphy, K.; et al. A Review of Relational Machine Learning for Knowledge Graphs: From Multi-Relational Link Prediction to Automated Knowledge Graph Construction. 03 2015.

[2] Auer, S.; Kovtun, V.; et al. Towards a Knowledge Graph for Science. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, WIMS '18, New York, NY, USA: Association for Computing Machinery, 2018, ISBN 9781450354899, doi: 10.1145/3227609.3227689. Available from: https://doi.org/10.1145/3227609.3227689

[3] About. Available from: https://wiki.dbpedia.org/about

[4] Interlinking. Available from: https://wiki.dbpedia.org/services-resources/interlinking

[5] Knowledge base. Jan 2020. Available from: https://en.wikipedia.org/wiki/Knowledge_base

[6] Ontology. Available from: https://wiki.dbpedia.org/services-resources/ontology

[7] Statistics. Available from: https://www.wikidata.org/wiki/Wikidata:Statistics

[8] Data Dumps — Freebase API (Deprecated) — Google Developers. Available from: https://developers.google.com/freebase

[9] YAGO: A High-Quality Knowledge Base. Available from: https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

[10] FOAF (ontology). Sep 2019. Available from: https://en.wikipedia.org/wiki/FOAF_(ontology)

[11] Available from: https://dbpedia.org/sparql

[12] Hearst, M. A. Automatic Acquisition of Hyponyms from Large Text Corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*, 1992. Available from: https://www.aclweb.org/anthology/C92-2082

[13] Brin, S. Extracting Patterns and Relations from the World Wide Web. In *Selected Papers from the International Workshop on The World Wide Web and Databases*, WebDB '98, Berlin, Heidelberg: Springer-Verlag, 1998, ISBN 3540658904, p. 172–183.

[14] Agichtein, E.; Gravano, L. Snowball: Extracting Relations from Large Plain-Text Collections. In *In Proceedings of the 5th ACM International Conference on Digital Libraries*, 2000, pp. 85–94.

[15] Mintz, M.; Bills, S.; et al. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Suntec, Singapore: Association for Computational Linguistics, Aug. 2009, pp. 1003–1011. Available from: https://www.aclweb.org/anthology/P09-1113

[16] What is Machine Learning? A definition. Nov 2019. Available from: https://expertsystem.com/machine-learning-definition/

[17] Haykin, S. S. *Neural networks and learning machines*. Upper Saddle River, NJ: Pearson Education, third edition, 2009.

[18] Available from: https://persistence.uni-leipzig.org/nlp2rdf/

[19] General Entity Annotator Benchmark. Available from: http://gerbil.aksw.org/gerbil/

[20] Entity Linking. Available from: http://nlpprogress.com/english/entity_linking.html

[21] DBpedia Spotlight Shedding light on the web of documents. Available from: https://www.dbpedia-spotlight.org/

[22] Coreference. Dec 2019. Available from: https://en.wikipedia.org/wiki/Coreference

[23] The Stanford NLP Group. Available from: https://nlp.stanford.edu/projects/coref.shtml

[24] TensorFlow Core. Available from: https://www.tensorflow.org/overview

[25] About Keras models. Available from: https://keras.io/models/about-keras-models/

[26] PyTorch documentation. Available from: https://pytorch.org/docs/stable/index.html

[27] Amaiya. amaiya/ktrain. Dec 2019. Available from: https://github.com/amaiya/ktrain

[28] Mikolov, T.; Grave, E.; et al. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[29] Google-Research. google-research/bert. Oct 2019. Available from: https://github.com/google-research/bert

Appendix A

Acronyms

NIF Graphical user interface

KB Extensible markup language

YAGO Graphical user interface

TTL Extensible markup language

CNN Convolutional neural network

ANN Extensible markup language

RELU Rectified linear unit

RDF Resource Description Framework

JSON-LD JSON line data

XML Extensible markup language

VM Virtual Machine

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `data` ..... the directory with executables
- `src` ..... the directory of source codes
  - `WikiExtraction` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format