Technische Universitat Dresden

International MSc Program in
Computational Logic (MCL)

Institute for Artificial Intelligence

"Predicting relation targets of DBPedia properties using vector representations from word2vec"

Student:
Happy Rani Das

Supervisor:
Prof. Dr. Sebastian
Rudolph

April 7, 2018

Abstract

Vector representation of words has been learned by many method, word2vec is one of them and useful in many natural language processing and information retrieval. Though, surprising fact is that Vector representation of words have not been applied for Dbpedia data to aggregate semantic information stored in word embeddings to predict dbpedia properties . In this paper, I am focusing on how Vector representation of words from word2vec can be applied for Dbpedia properties in order to get missing information. The main goals of Predicting relation targets of DBPedia properties using vector representations from word2vec are to find corresponding resources from the Dbpedia and to apply Word2Vec technique to find missing information.

1 Introduction

1.1 Background

Word2vec is a neural network, which takes text corpus as input and produces a set of vectors as a result. In the beginning word2vec build a vocabulary from a large text corpus and then produces group of vectors. It first constructs a vocabulary from the training text data and then learns vector representation of words. There are two ways to represent word2vec model architecture 1. continuous bag-of-words (predicts a missing word given a window of context words or word sequence) and 2. skip-gram (predict the neighboring window of target context by using a word) These word2vec model architecture used in machine learning areas, natural language processing and advance research areas. [1].

Continuous bag-of-words (CBOW) and continuous skip-gram model architecture are very popular nowadays in the machine learning areas and further research. Another depiction of words is dense vector came to know by word2vec. Dense vector have exceptionally been displayed to demonstrate same sense and it is beneficent in the immense range from data analytics to natural language processing.

As an example, words that have equivalent explanation will have analogous vectors because of cosine similarity and the words whose doesn't have equivalent explanation will have unalike vectors. It is quite surprising that, word vectors follow the likeness rule. For instance, presume the likeness

"Berlin is to Germany as Paris is to France". It gives us the result like following

$$v_{Germany} - v_{Berlin} + v_{Paris} = v_{France}$$

where $v_{Germany}; v_{Berlin}; v_{Paris} and v_{France}$ are the word vectors for Germany, Berlin, Paris, and France respectively. [2].

1.2 Project Description

This project is focused on the Extracting Vector Representation of DBPedia Properties from word2vec. Word2Vec has been applied in several areas with the purpose of detect similarity and to find out nearest word. However, Word2Vec has not been applied for DBPedia properties. The main goal of this project is to introduce techniques that can be used for learning DBPedia data and to find out corresponding missing information from DBPedia. DBpedia ("DB" stand for "database") is a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web [defined by http://wiki.dbpedia.org/about].

If the user has two sentences like-

1. Berlin is the capital
2. Paris is the capital.

The result of the Word2vec similarity will be the words ending up near to one another. Suppose, if we train a model with (input:Berlin,output:Capital) and (input:Paris,output:Capital) this will eventually give insight the model to understand that, Berlin and Paris both as connected to capital, thus Berlin and Paris closely in the Word2Vec similarity.

The Extracting Vector Representation of DBPedia Properties from word2vec is developed in python which takes DBPedia properties as input and returns nearest or similar missing information as output.

2 Related Work

3 Methodology

Figure 1 shows the flowchart diagram of our project. Dbpedia data is given as input which is then passed into pre-trained model GoogleNews-vectors-

negative300.bin. then we get the vector representation of the given data. Thereafter we try to find out the nearest word based on the given data. After that we try to figure out the missing information like if one word is related to another word, the same type of word would be related to other word which we don't know about it. For instance, if Donald Trump is to republican as Barack Obama is to what? And the output would be Democratic.

3.1 Query DBPedia Properties for relations

This is most important and tricky part in this project. Properties is a connections between objects in DBPedia. To get all of the objects for specified properties we use SPARQL query language. For our project we think about four SPARQL query against DBPedia.

Query 1:

The following code shows how to execute SPARQL query against DBpedia for properties 'country' and 'capital' in order to get all of the country and capital list :

1. SELECT DISTINCT ?country ?capital WHERE

2.{

3. ?city rdf:type dbo:City ;

4. rdfs:label ?label ;

5. dbo:country ?country .

6.?country dbo:capital ?capital .

7.} order by ?country

Table: 1 displayed the result of query: 1.

Query:2

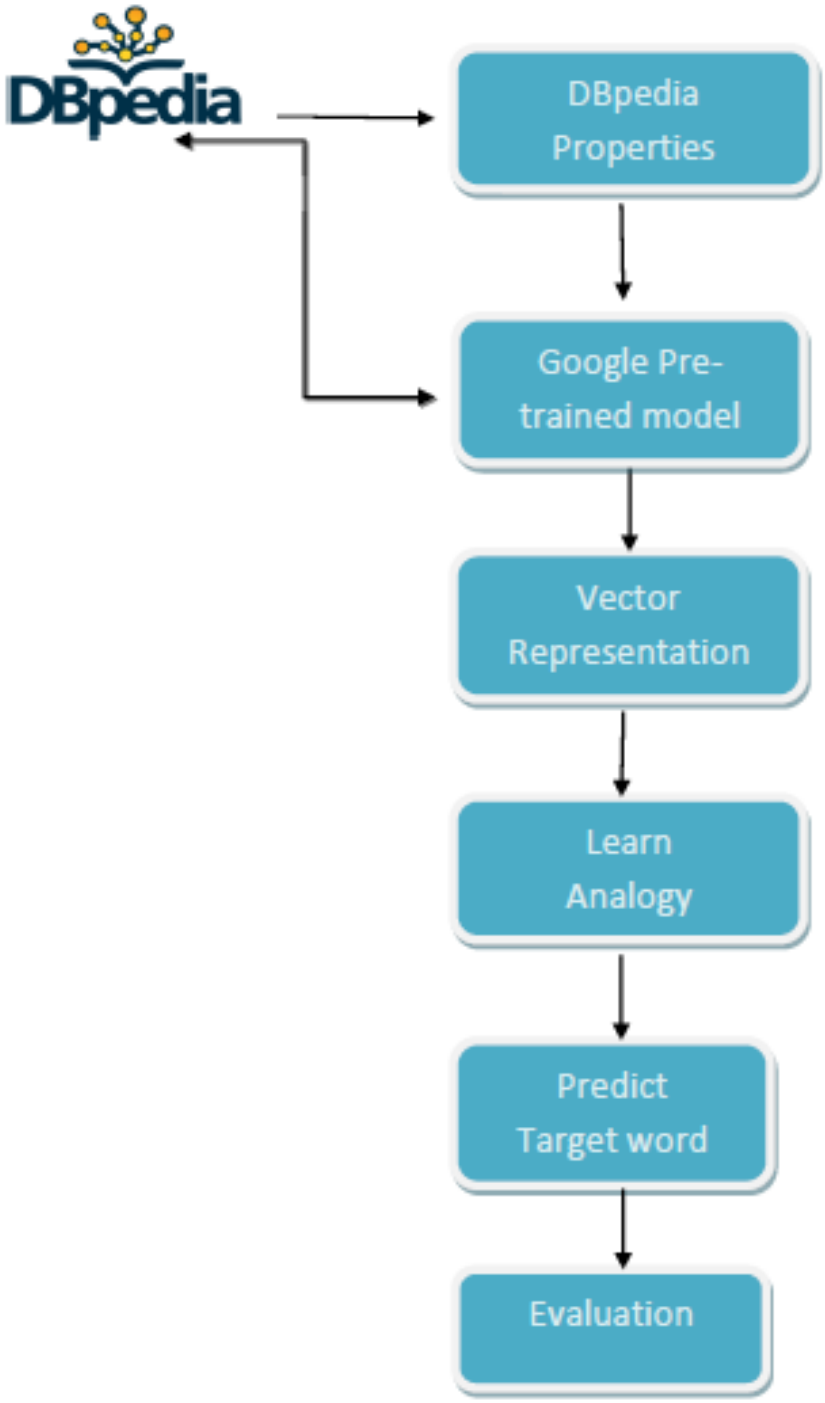

Figure 1: Figure 1: The General Scheme of Extracting Vector Representation of DBpedia Properties from Word2Vec

country	capital
http://dbpedia.org/resource/Afghanistan	http://dbpedia.org/resource/Kabul
http://dbpedia.org/resource/Algeria	http://dbpedia.org/resource/Algiers
http://dbpedia.org/resource/Angola	http://dbpedia.org/resource/Luanda
http://dbpedia.org/resource/Argentina	http://dbpedia.org/resource/Buenos_Aires
http://dbpedia.org/resource/Armenia	http://dbpedia.org/resource/Yerevan
http://dbpedia.org/resource/Australia	http://dbpedia.org/resource/Canberra
http://dbpedia.org/resource/Austria	http://dbpedia.org/resource/Vienna
http://dbpedia.org/resource/Azerbaijan	http://dbpedia.org/resource/Baku
http://dbpedia.org/resource/Bahrain	http://dbpedia.org/resource/Manama
http://dbpedia.org/resource/Bangladesh	http://dbpedia.org/resource/Dhaka
http://dbpedia.org/resource/Barbados	http://dbpedia.org/resource/Bridgetown
http://dbpedia.org/resource/Belarus	http://dbpedia.org/resource/Minsk
http://dbpedia.org/resource/Belgium	http://dbpedia.org/resource/City_of_Brussels
http://dbpedia.org/resource/Belize	http://dbpedia.org/resource/Belmopan
http://dbpedia.org/resource/Benin	http://dbpedia.org/resource/Porto-Novo
http://dbpedia.org/resource/Bolivia	http://dbpedia.org/resource/Sucre
http://dbpedia.org/resource/Bosnia_and_Herzegovina	http://dbpedia.org/resource/Sarajevo
http://dbpedia.org/resource/Brazil	http://dbpedia.org/resource/Brasília
http://dbpedia.org/resource/Bulgaria	http://dbpedia.org/resource/Sofia
http://dbpedia.org/resource/Burkina_Faso	http://dbpedia.org/resource/Ouagadougou
http://dbpedia.org/resource/Burundi	http://dbpedia.org/resource/Bujumbura
http://dbpedia.org/resource/Cambodia	http://dbpedia.org/resource/Phnom_Penh
http://dbpedia.org/resource/Cameroon	http://dbpedia.org/resource/Yaoundé

Table 1: Output for the input properties country and capital.

The subsequent query applies for properties country and currency and returns all of the results against those properties:

1. SELECT DISTINCT ?country ?currency WHERE

2.{

3. ?city rdf:type dbo:City ;

4. rdfs:label ?label ;

5. dbo:country ?country .

6.?country dbo:currency ?currency .

7.} order by ?country

Query: 3

Here is a query which returns all of the persons and their corresponding party name under the properties Person and party

1. SELECT DISTINCT ?person ?party WHERE

2.{

3. ?city rdf:type dbo:City ;

4. ?person rdf:type dbo:Person .

5.?person dbo:party ?party.

6.} order by ?person

For SPARQL if the result is within 40,000 it is possible to shown once in a time. But if the result is more than 40,000 it's not possible to display in one

page once at a time. Because The DBpedia SPARQL endpoint is configured in the following way:

$$\text{MaxSortedTopRows} = 40000.$$

In DBpedia for dbo:Person and dbo:party there are huge amounts of data. In order to get the rest of the data we use offset which allows to get the next results from the offset index. For instance, we consider the following query and append it into the result:

1.SELECT DISTINCT ?person ?party WHERE

2.{

3.?person rdf:type dbo:Person .

4.?person dbo:party ?party.

5.} order by ?person

6. offset 43880 .

Query: 4

The subsequent query return all of the results for the property spouse.

1.SELECT DISTINCT ?x ?y WHERE

2.{

3.?x dbo:spouse ?y.

4.?y dbo:spouse ?x.

5.} order by ?x

3.2 Retrieved data clean

After retrieving all data from DBpedia we try to clean them . Such as we remove "http://dbpedia.org/resource/" from all of the data, if the result contain "-" we replace it "_", in case data are inside the parentheses(()) we take aside them with parenthesis and at the end of line if there is "_" we carry away them from the line in order to make it meaningful for further processing.

3.3 Google's pre-trained model

We are considering google pre-trained word vector model (GoogleNews-vectors-negative300.bin) to represents word vector. Google pre-trained word vector model is published by Tomas Mikolov and his team. It contains 3 million words and phrases and from a Google News dataset they trained around 100 billion words [4]. We passed retrieved cleaned data into Google's pre-trained model. If the data is in the model it returns the output (filtered data) otherwise it does not return the data which are not in the model. The output we are using them for our project experiment.

3.4 Vector Representation

References

[1] https://code.google.com/archive/p/word2vec/

[2] http://www.1-4-5.net/ dmm/ml/how_does_word2vec_work.pdf

[3] viewexportask othersask others Philipp Heim, Sebastian Hellmann, Jens Lehmann, Steffen Lohmann, Timo Stegemann: RelFinder: Revealing Relationships in RDF Knowledge Bases. SAMT 2009: 182-187

[4] http://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/