TECHNISCHE UNIVERSITÄT DRESDEN

FACULTY OF COMPUTER SCIENCE
INSTITUTE OF ARTIFICIAL INTELLIGENCE
CHAIR OF COMPUTATIONAL LOGIC

Master of Science

Project Report

Predicting relation targets of DBpedia properties using vector representations from word2vec

Happy Rani Das

Supervisor: Dr. Dagmar Gromann

Dresden, August 19, 2018

Abstract

Vector representation of word has been learned by many methods, word2vec is one of them and used in many natural language processing and information retrieval to detect word similarity, analogical reasoning and so on. However, vector representation of word has been applied for DBpedia data regarding the relational task, but it has not been yet applied for DBpedia data to aggregate semantic information stored in word embeddings. This project targets to increase the accuracy of analogical reasoning by aggregating semantic information of DBpedia data to predict the relation targets. Therefore, we introduced two super vectors, which are capable to accumulate several vectors in the analogy task and which can incredibly increase the accuracy of the analogous characteristic of the vectors. The following four relations namely capital-country, currency-country, person-party and company-headquarter are considered from DBpedia for this study to evaluate the system performance. With the increasing number of vectors, the performance has increased for all of the above-described relations and the highest performance was achieved by the company-headquarter relation.

Contents

List of Figures II

List of Tables III

1 Introduction 1

2 Preliminaries and Related Work 4
- 2.1 Preliminaries 4
- 2.2 Related Work 6

3 Methodology 9
- 3.1 Query DBpedia Properties for relations 9
- 3.2 Cleaning 13
- 3.3 Google's pre-trained model 13
- 3.4 Vector Representation 14
- 3.5 Learn Analogy 14
- 3.6 Predict Relation Target 15
- 3.7 Evaluation 16

4 Results 18

5 Discussion 28

6 Conclusion and Outlook 30

Bibliography 31

List of Figures

2.1 The parallelogram model for the analogy boy : girl :: man : ? 6

3.1 The General Scheme of Extracting Vector Representation of DBpedia Properties from Word2vec 10
3.2 Most similar word of man 14

4.1 Average accuracy against vector number for capital and country 21
4.2 Average accuracy against vector number for currency and country 23
4.3 Average accuracy against vector number for person and party 25
4.4 Average accuracy against vector number for company and headquarter . 25
4.5 Average accuracy against vector number for the four relations 27

List of Tables

3.1 Output for the input properties country and capital. 11
3.2 Extracted and filtered data for four relations 16

4.1 Average accuracy for properties country-capital 19
4.2 Average accuracy for properties country-currency 22
4.3 Average accuracy for properties person-party 24
4.4 Average accuracy for properties company-headquarter 26

1 Introduction

Word2vec is a proficient method in natural language modeling, 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 a group of vectors. There are two ways to represent word2vec model architecture [Mikolov et al. 2013].

1. Continuous bag-of-words (predict a missing word from a given window of context words or word sequence) and
2. Skip-gram (predict the neighboring window of target context by using a word)

Continuous bag-of-words (CBOW) and continuous skip-gram model architecture are very popular nowadays in the machine learning areas, natural language processing and advance research areas.

Words that have equivalent meaning will have analogous vectors and the words whose doesn't have equivalent explanation will have unalike vectors. It is quite surprising that, word vectors follow the analogy rule. For instance, presume the analogy "man is to king as woman is to queen". It gives the following results.

$$v_{king} - v_{man} + v_{woman} = v_{queen} \tag{1.1}$$

where v_{king}, v_{man}, v_{woman} and v_{queen} are the word vectors for king, man, woman and queen respectively.

Word2vec has been applied in many areas with the objective of generating or extracting information to solve a specific problem from the specific knowledge base. Different types of knowledge bases are available. DBpedia is one of them. DBpedia ("DB" stand for "database") is a crowd-sourced community effort to extract structured information from

Wikipedia, make this information available on the web and has been used as a dataset for diverse purposes[1].

The intention of this project is to introduce a technique that can be used for learning DBpedia data and to find out corresponding relation targets from DBpedia. 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 gives insight the model to understand that, Berlin and Paris both are connected to capital, thus Berlin and Paris closely related in the word2vec similarity.

The main aim of this project is the implementation of a technique that can aggregate semantic information stored in word embeddings to predict DBpedia properties. The idea behind the technique is based on a new system; launched to increase the accuracy which is capable to amalgamate more than one vector. Increasing number of vectors should work better than single vector because vector addition of same relation type should strengthen the characteristic representation. For example, if two vectors such as male, female in a analogical reasoning in equation 1.1 for the relation gender provide a good basis, then theoritically combining more vector of the same category in a relation should improve the results. For instance, man, boy, husband, father etc. should provide a stronger representation for the characteristic “male” than just man.

Vector representation of words have not been applied for DBpedia data to aggregate semantic information stored in word embeddings. In this project we introduce a technique to aggregate semantic information.

The prediction of relation targets from DBpedia properties using vector representations is developed in python which takes DBedia properties as input and returns nearest words or relation targets as output.

[1]http://wiki.dbpedia.org/about

The rest of this report is structured as follows. An overview of preliminaries and related work is discussed in Chapter 2. In the following Chapter, the methodology of the project is described. The results of the project is summarized in Chapter 4 and they are discussed in Chapter 5 subsequently. Finally, the report is concluded with Chapter 6 along with future possibility of the work.

2 Preliminaries and Related Work

2.1 Preliminaries

2.1.1 SPARQL Query Language

SPARQL is a semantic query language to query RDF graph. It is pronounced as "sparkle", a recursive acronym stands for SPARQL protocol. SPARQL is capable to retrieve and manipulate information which is stored in a Resource Description Framework (RDF) format. There are diverse types of output format available for SPARQL such as result sets, JSON, RDF/XML, CSV etc. A SPARQL is mainly based on the basic graph/ triple pattern matching stored in the RDF graph where it tries to find out the set of triples from the RDF graph [Morsey et al. 2012]. The following example shows how SPARQL query looks like.

Example: SPARQL query -

1. PREFIX dbo: ⟨ http://dbpedia.org/ontology/⟩

2. PREFIX dbr: ⟨ http://dbpedia.org/resource/⟩

3. PREFIX s: ⟨ http://schema.org/⟩

4. SELECT * WHERE {

5. ?Hotel a s:Hotel.

6. ?Hotel dbo:location dbr:Dresden.

7.}

The above query is a combination of prefixes and triples. Prefixes are shorthand for long URIs. The SPARQL query returns all the hotels in Dresden as a result when it executed against DBpedia. For example, Dresden has only two hotel under dbo:location and they are:-

1. "http://dbpedia.org/resource/Taschenbergpalais" and
2. "http://dbpedia.org/resource/Swissôtel_Dresden_Am_Schloss"

2.1.2 Resource Description Framework (RDF)

The Resource Description Framework (RDF) is a general-purpose language and standard model for data interchange on the Semantic Web [Lassila, Swick, et al. 1998]. It has URLs, URIs and IRIs to uniquely identify resources on the web. As an example, http://dbpedia.org/resource/Donald_Trump is globally unique. A Simple syntax for RDF is Turtle (Terse RDF Triple Language) specified by a W3C recommendation. RDF is the building block of the Semantic Web, made up of triple of the form where a triple is consists of subject(resource or blank node), properties(resource) and object(resource, literal or blank node). An example of RDF triple is:-

dbr:Barack_Obama dbp:spouse "Michelle_Obama"

where dbr:Barack_Obama is subject, dbp:spouse is predicate/properties and Michelle_Obama is object.

2.1.3 Word embeddings

Word embeddings are the central to natural language processing and efficient method to capture inner words semantics [Levy and Goldberg 2014]. a Word embedding is a vector representation of a word where each word from a vocabulary is mapped to a vector. Vector representation of word plays an increasingly essential role in predicting missing information by capturing semantic and syntactic information of words, and also this representation can be useful in many areas such as question answering, information retrieval, text classification, text summarization and so on [Teofili 2017]. To get vector representation of a word we are using google pre-trained word vector model.

2.2 Related Work

Vector representation of word has gained enormous attention in the field of machine learning. Several techniques are introduced to get word vector. Word2vec and GloVe has gained tremendous popularity to predict semantic similarity.

Chen et al. [2017] proposed a method, which is useful to detect similarity in a syntactic way for a set of objects and their relationship. The relationship between one pair of objects is to that of another pair of objects. This is done by Parallelogram model of analogy [2017]. Parallelogram model of analogy is the representations of objects that contain data which is essential to presume relationship between objects (see fig. 2.1), where objects are represented as points and relations between objects are represented by their difference vectors.

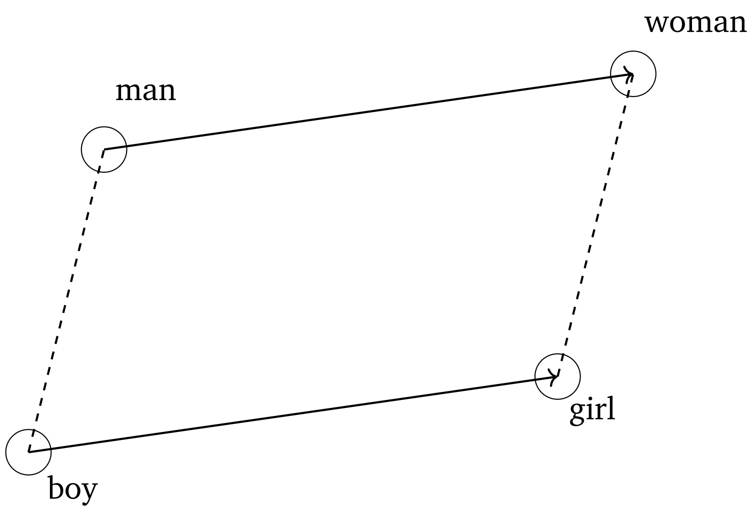


Figure 2.1: The parallelogram model for the analogy boy : girl :: man : ?

Based on the parallelogram model, two words pairs (s_1, t_1) and (s_2, t_2) are relationally similar if their vector differences $(vect_1 - vecs_1$ and $vect_2 - vecs_2)$ are similar. Different types of methods are available to measure the vector difference similarity. Cosine similarity is one of them. The cosine similarity is a measure of two non-zero vectors that computes the cosine of the angle between them[1]. Suppose, a= $vect_1 - vecs_1$ and b = $vect_2 - vecs_2$. The calculation of cosine similarity is given in the following.

$$\frac{a.b}{||a||||b||} \tag{2.1}$$

Yoshua Bengio et al. [2003] introduced a technique for learning the distributed representations of words which allows to make semantically new sentences from each training sentence. In their approach, they represented word as a distributed feature vector and a new sentence was possible to make from a training data set. For example, a sentence " A tiger is running in the jungle" helps to make another sentence "The fox was walking in a forest".

Andriy Mnih and Koray Kavukcuoglu [2013] proposed an approach to learning word embeddings based on training lightweight language models. In their paper, they focused on the analogy-based questions sets, which has the form “x is to y as z is to ”, defined as x : $y \to z$: ? . The idea was if x is related to y, then the same way z is related to what? Their target was how efficiently it could detect the fourth word and they proved their approach produced better word embeddings performance.

Zhiwei Chen et al. [2017] proposed a methodology to deduce the semantic relations in word embeddings from WordNet and Unified Medical Language System (UMLS). They trained multiple word embeddings from Wikipedia. In the field of text mining and Natural Language Processing (NLP), word embedding has been exhibited efficiently for capturing syntactic and semantic relations in the past few years. To get the word embeddings, they used three tools; word2vec, dependency-based word embeddings, and GloVe. The nine semantic relations synonym, antonym, hypernym, hyponym, holonym, meronym, sibling, derivationally related forms and pertainym are explored by them. They usedwordNet and Unified Medical Language System (UMLS) resources. Wordnet unites nouns, verbs, adjectives, and adverbs into sets of cognitive synsets. Synsets are a combination of lexical and semantic relations. Unified Medical Language System is a synopsis of numerous controlled vocabularies in the biomedical sciences. They constructed a standard database with WordNet and UMLS to evaluate the performance of nine semantic relations. Pertainym relation acquired the maximum retrieved ratio to

[1]https://en.wikipedia.org/wiki/Cosine_similarity

find nearest neighbors. They received better performance with Word2Vec than GloVe for almost all of the semantic relations. On the other hand, dependency-based word embeddings obtained poor performance than Word2Vec and GloVe for the relations pertainym, derivation, meronym and holonym.

The four linguistic relations inflectional, derivational, lexicographic and encyclopedic semantics are introduced by Gladkova et al. [2016]. An analogy is a successful word to detect diverse types of semantic similarity. Their target was to know which types of analogies will be able to work perfectly and which are not. For that they have tested 99200 questions in 40 relations. For word embedding, they used GloVe and SVD method. In their experiments, some relation has given excellent accuracy and some didn't and the accuracy varied between around 10% to about 98%. In one hand, they received around 98% accuracy for the relation capital-country. On the other hand, the accuracy is around 10% for the relation meronyms–member. This proves that a model may be more effective with some categories rather than others. The reason is that some relations may be not captured nicely with word embedding or vector offset. They have shown that the accuracy is also varied with window-size, word-category and vector dimensionality. GloVe gives better performance than SVD. Out of 40 relations, 13 relations achieved more than 40% accuracy for GloVe and SVD acquired more than 40% accuracy only for six relations.

3 Methodology

Figure 3.1 shows the flowchart diagram of the project. DBpedia data is considered as input and pre-trained model GoogleNews-vectors-negative300.bin is used to acquire the vector representation of the given DBpedia data. Thereafter we tried to find out the nearest word based on the given data. After that, the relation target was figured out as if one word is related to another word in a relational task, the same relational type of another word would be related to other word which is unknown to us. 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

DBpedia has huge amounts of data. It is a combination of resources, ontology, properties and many more. In the RDF preliminaries, we have seen predicate represents the properties or relation for RDF triple. Properties are used to link entities together. To get all of the resources for specified properties we used SPARQL query language. For this project, we consider four SPARQL query against DBpedia.

Query 1:

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

1. SELECT DISTINCT ?country ?capital

WHERE

2. {

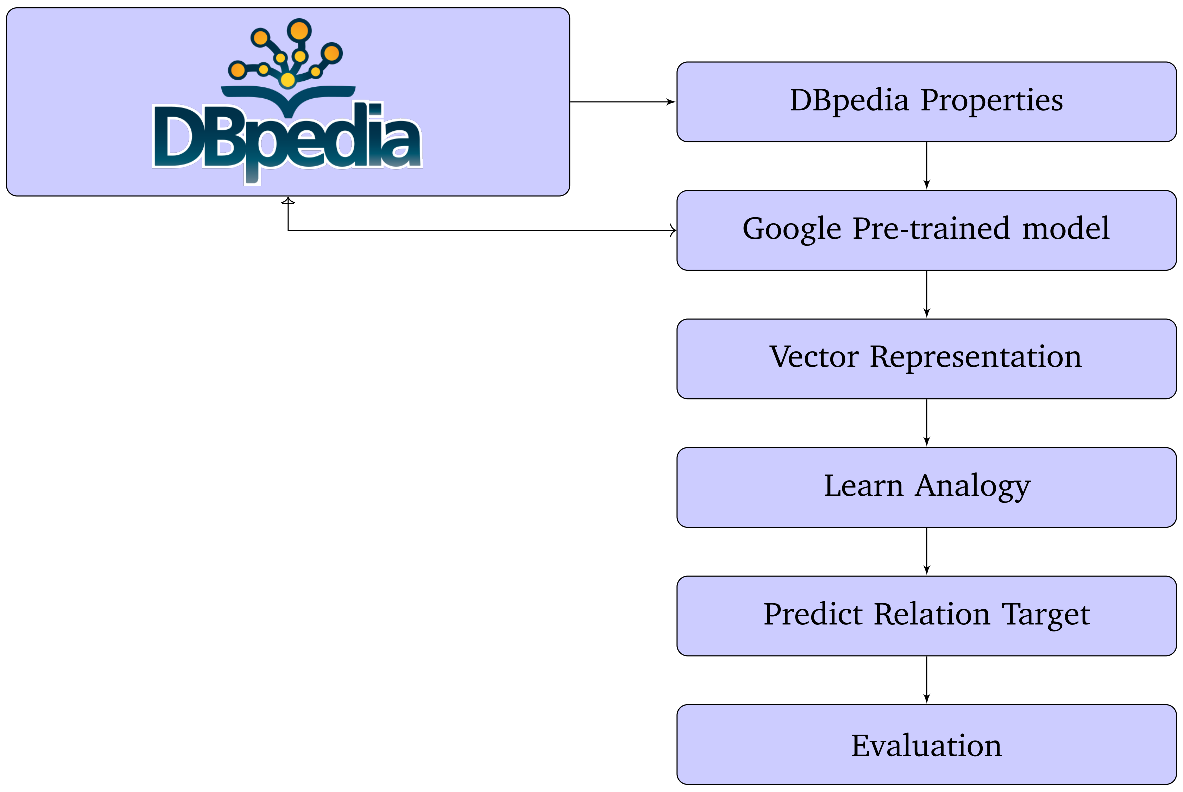


Figure 3.1: The General Scheme of Extracting Vector Representation of DBpedia Properties from Word2vec

3. ?city rdf:type dbo:City;

4. rdfs:label ?label;

5. dbo:country ?country.

6. ?country dbo:capital ?capital.

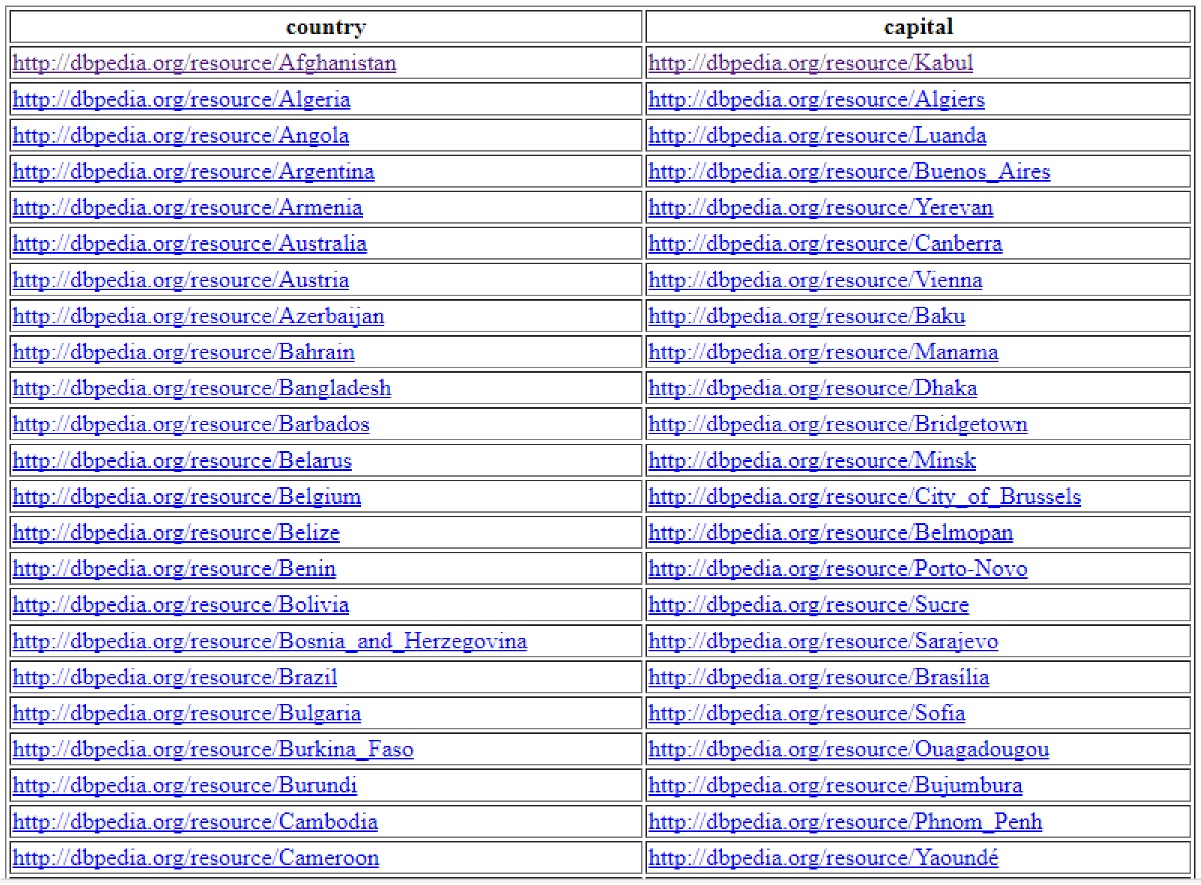
7. } order by ?country

Table3.1 displayed the result of query 1.

Query: 2

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

1. SELECT DISTINCT ?country ?currency WHERE



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 3.1: Output for the input properties country and capital.

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

```
SELECT DISTINCT ?person ?party WHERE
{
?city rdf:type dbo:City;
?person rdf:type dbo:Person.
?person dbo:party ?party.
} order by ?person
```

For SPARQL if the result is within 40000 it is possible to show once at a time. But if the result is more than 40000 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 results:

```
SELECT DISTINCT ?person ?party WHERE
{
?person rdf:type dbo:Person.
?person dbo:party ?party.
} order by ?person.
offset 43880.
```

Query: 4

The subsequent query executed against DBpedia to return all of the results for the properties company and headquarter.

```
select DISTINCT ?x ?y where
```

2. {

3. ?x a dbo:Company.

4. ?y a dbo:City.

5. ?x dbo:headquarter ?y.

6. } order by ?x

3.2 Cleaning

After extracting all the data from DBpedia we cleaned them. For instance, we removed "http://dbpedia.org/resource/" from all of the data. If the result contains "-" we replaced it with "_". When the data are inside the parentheses (()) we took aside them with parenthesis and at the end of the line if there is "_", we carried away them from the line in order to make it meaningful for further processing. Thereafter we used google pre-trained word vector model to retrieve those data which are in the word vector model. For example, if the DBpedia data are in the pre-trained model it returns the output (filtered data) otherwise it does not return the data which are not in the model. As we are using google pre-trained word vector model to get vector representation of a word. So, we had to make sure that the DBpedia data are in the google pre-trained word vector model.

3.3 Google's pre-trained model

We are considering google pre-trained word vector model (GoogleNews-vectors-negative300.bin) to represents word as a vector. Google pre-trained word vector model is published by Mikolov et al. It contains three million words and phrases and from a Google News dataset they trained around 100 billion words[1]. We retrieved vectors for words from Google's pre-trained model for our project experiment.

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

```
[('woman', 0.7664012312889099),
('boy', 0.6824870109558105),
('teenager', 0.6586930155754089),
('teenage_girl', 0.6147903800010681),
('girl', 0.5921714305877686)]
```

Figure 3.2: Most similar word of man

3.4 Vector Representation

Representation of vector for word relies on many techniques such as GloVe, Word2vec and so on. We consider word2vec in order to get vector representation of the word. The concept behind word to vector has many advantages. For example

1. Semantic likeliness (similarity) of words is represented by vector.
2. Analogical task is also represented by vector.

Vector for word is retrieved from the Google's pre-trained model. The number of dimensions for a vector is fixed and it is usually 300.

3.5 Learn Analogy

One of the advantages of the vector representation of word is similarity calculation. It's possible to find out the most similar word and their distance by using word vector. For example, if we enter man as an input it returns the following words and the corresponding distance from Google's pre-trained model as shown in Figure 3.2. Analogy has been playing a significant role to detect similarity in a syntactic and semantic way for a set of words and their relationship. Learn analogy is the way to measure the relational similarity to know which word vector contains semantic information. For a given group of word pair source:target the intention of learn analogy is to predict the target word pair which is having the same relation. As an example for a given word pair Berlin:Germany the target word pair would be Paris:France, Venice:Austria, Madrid:Spain and so on.

3.6 Predict Relation Target

It is also possible to predict relation target by using vector representation from word2vec model. The relationship between the two words looks like $w_1 \rightarrow w_2$. And the target word would be $w_3 \rightarrow w_4$?. So, we need to predict the relation target w_4? For our approach, in order to predict the relation target and to get the more accurate results, we implemented two super vectors: super vector source(SVS) and super vector target(SVT). Super vector source (SVS) and super vector target(SVT) are the amalgamation of one to twenty relation sources and relation targets respectively from the training data.

Based on the equation 1.1, we can write-

$$1\times \text{ testing source - Vec([SVS]) + Vec([SVT]) = Result}$$

Where Vec([SVS]) and Vec([SVT]) are the combination of relation sources and relation targets respectively from the training data. Result is the prediction of testing target for a respective testing source. For obtaining relation target of testing data, the relation source of testing data is composed with the SVS and SVT. We have tested each individual data pair from the test set in the same way and evaluate the obtained prediction against the target value of the test set.

Assume, if Kigali is related to Rwanda, Paris is to France, Vienna is to Austria, Lima is to Peru, then Kabul is related to what? To obtain the result of Kabul is related to what, we aggregated all of the capital names together to construct SVS and country names together to build SVT. For example, consider the following representation:

Vec["Kabul"] - ((vec["Kigali"] + vec["Paris"] + vec["Vienna"] + vec["Lima"])) +
vec["Rwanda"] + vec["France"]+ vec["Austria"] + vec["Peru"] = ?

Where Vec[“Kabu”]l is the testing source. Super vector source (SVS) and super vector target (SVT) are the composition of all the capital name and country name respectively in vector form. Output would be the predicted results against the testing source Kabul.

3.7 Evaluation

For this project, the following four relations capital-country, currency-country, person-party and company-headquarter are chosen from DBpedia to evaluate the system performance. At the beginning, data were extracted from DBpedia by using SPARQL for the above four relations. Thereafter we have done cleaning process and obtained filtered data to make sure that the data are meaningful for further processing. We are using those filtered data for the next steps. The (filtered) data are divided into two parts:

1. For training
2. For testing

In every DBpedia properties we have taken randomly half or more than half filtered data for testing and the rest are for training. Table 3.2 is listed below shows how many data were extracted from DBpedia for the above four relations. Moreover, it captured the filtered data too, which are divided into training and testing set.

Table 3.2: Extracted and filtered data for four relations

Properties	**Extracted Data**	**Filtered Data**	
		Testing	**Training**
Person-Party	86851	3400	1430
Country-Capital	182	73	73
Country-Currency	182	33	32
Company-Headquarter	2330	122	83

We have tested all of the relation sources from the testing set against the training dataset in Google's pre-trained model to get relation targets of the testing set. From training set, we considered maximum 20 vector set(source, target). At first, we took one vector set (source, target) randomly to generate super vector source and super vector target, tested against all of the relation sources from the testing set and received predicted relation targets. We counted the number of times the test target is predicted correctly. For one vector set(source, target), data were taken randomly 10 times from the training set and in the same way super vectors were formed. In every time we counted how many correct results were there to calculate accuracy. A correct result is the sum of all the predicted correct testing targets for testing sources. Then we divided the correct results with the total number of testing targets to acquire the accuracy. Accuracy is the ratio of the number of the correct results to the total number of testing targets. We repeated it 10 times randomly in the same way and calculated accuracy respectively. Afterward, we average these ten accuracies to achieved the average accuracy.

Average accuracy is = average(accuracy1, accuracy2, accuracy3,......, accuracy10)

Secondly, two vector set (source, target) were chosen and applied the same procedure to get average accuracy. After that, we proceed consecutively till twenty in order to evaluate the performance of the project and to observe which vector set is giving the highest accuracy.

4 Results

After constructing super vectors correct item, vector number, accuracy and average accuracy were calculated to evaluate the system's performance. At the beginning, correct item is set to zero to calculate the accuracy. A correct item is the sum of all the items, which predict the correct test targets for testing data. If the first predicted relation target is correct, the correct item is incremented from zero to one. We repeated the process for the entire testing set and calculated correct item. Vector number is the number of (source, target) pair from the training dataset. Accuracy is the ratio of the number of the correct items to the total number of test targets whereas average accuracy is the ratio of the accumulated accuracies to the number of obtained accuracies. The accuracy and average accuracy are calculated using the formulas in Equations 4.1 and 4.2 respectively.

$$Accuracy = Number\_of\_Correct\_Items/Total\_Number\_of\_Test\_Targets \tag{4.1}$$

$$Average\_Accuracy = Accumulated\_Accuracies/Number\_of\_Obtained\_Accuracies \tag{4.2}$$

Capital-Country

For properties, capital and country the following average accuracy is acquired for the consecutive vector number as shown in Table 4.1. The accuracy is obtained by calculating the number of correct items to the total number of test targets. We consider the top three results. If the correct answer is in the top three results the correct item will be incremented. For properties, capital-country the testing dataset contains 73 total items. The relation sources are capital here and its predicted country as a relation target. The dataset was tested against the training dataset. When the vector number was 18 the correct items were 70 for the first iteration and the accuracy is:-

1. Accuracy = (70/73) = 0.958904109589041

Table 4.1: Average accuracy for properties country-capital

Number of Vectors	Average Accuracy	Percentage
1	0.797260274	79.7260274
2	0.884931507	88.49315068
3	0.9	90
4	0.915068493	91.50684932
5	0.902739726	90.2739726
6	0.923287671	92.32876712
7	0.923287671	92.32876712
8	0.905479452	90.54794521
9	0.938356164	93.83561644
10	0.92739726	92.73972603
11	0.932876712	93.28767123
12	0.932876712	93.28767123
13	0.924657534	92.46575342
14	0.92739726	92.73972603
15	0.938356164	93.83561644
16	0.930136986	93.01369863
17	0.926027397	92.60273973
18	0.94109589	94.10958904
19	0.934246575	93.42465753
20	0.926027397	92.60273973

We have discussed in the evaluation that for every vector set we have taken the data randomly 10 times from the training dataset. As the data were chosen randomly, so accuracy was changing in every time and results are follows.

2. Accuracy = (68/73) = 0.9315068493

3. Accuracy = (68/73) = 0.9315068493

4. Accuracy = (69/73) = 0.9452054795

5. Accuracy = (69/73) = 0.9452054795

6. Accuracy = (68/73) = 0.9315068493

7. Accuracy = (68/73) = 0.9315068493

8. Accuracy = (69/73) = 0.9452054795

9. Accuracy = (69/73) = 0.9452054795

10. Accuracy = (69/73) = 0.9452054795

The accumulated accuracies were calculated by aggregating the 10 accuracies, which is around 9.410958904109588. The average accuracy is:-

Average_Accuracy = (9.410958904109588/10) = 94.10958904109

As we can see from the results for properties capital and country in table-4.1 the highest average accuracy is around 94%. We observed that when the super vectors are for one vector, the average accuracy is 79.726%. But when we increased the vector number we received higher accuracy. The Average accuracy for two vectors is 88.493% and for 3 vectors is 90%. So, with the increasing number of vectors average accuracy is increasing. For 6 and 7 vectors, the average accuracy is same and it is 92.329% and for 8 vectors the average accuracy is 90.548%, which is a little bit less compared to vectors 6 and 7. The average accuracy is 93.836% for 9 vectors. We receive the highest average for vectors 18 and it is 94.110%. We noticed with the increasing number of vectors the average accuracy is increased or be stabled for consecutively one or two vectors and after that, it goes down around 1% for the next vector. But after the vector number of 18, it has started to walk toward a lower position. Since we are looking for higher accuracy, so we decided to make the super vectors with the number of 20 vectors.

The graphical visualization of the relation(capital, country) is shown in the Figure 4.1

Currency-Country

For properties currency and country we received the following average accuracy for the successive number of vector, shown in Table 4.2.The accuracy and the average accuracy was calculated in the same way as we calculated for the properties capital-country. Properties pair country-currency has 33 total items in the testing dataset. Countries name are considered as relation sources here and currencies name are the relation targets. For the correct items 21, the accuracy is:-

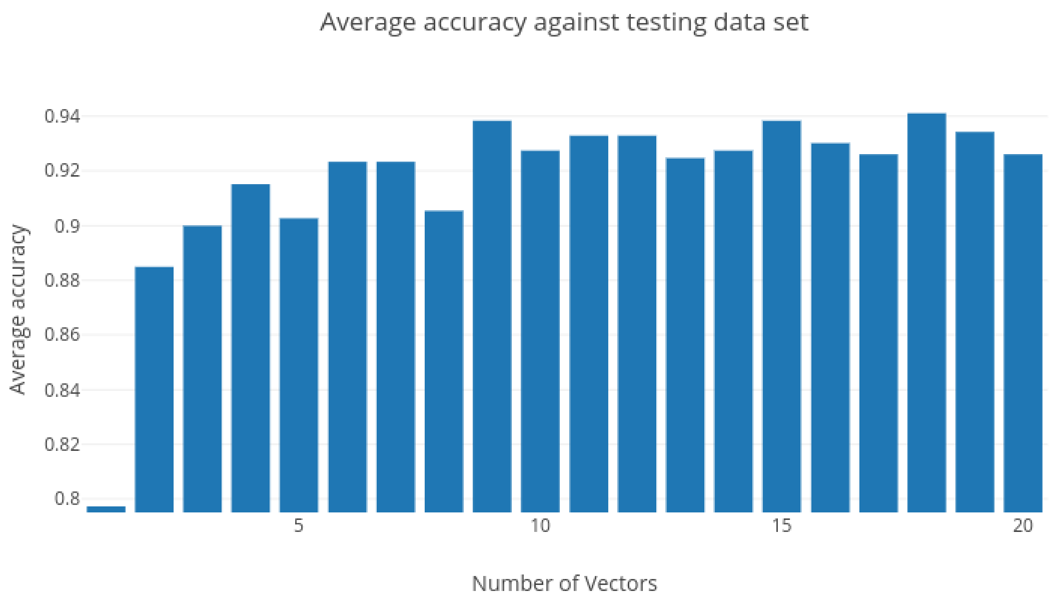


Figure 4.1: Average accuracy against vector number for capital and country

$$\text{Accuracy} = (21/33) = 0.6363636364$$

The highest average accuracy for properties currency and country is about 62%. We have marked from the table 4.2 that the average accuracy for one vector is 26.667%. we observed that the average accuracy is significantly increased for the vectors 2 and 3 and they are 43.939% and 50.909% respectively. After that, the average accuracy is getting up and a little bit down with the increasing number of vector. The average accuracy for vector 7 is 61.212%. Like the properties pair capital-country, the highest average accuracy for the properties pair currency-country is acquired by the vectors 18 and it is 62.121%.

The visualization of the relation(currency, country) is shown in the Figure 4.2

Person-Party

The following average accuracy is obtained by the consecutive vector number presents in Table 4.3 for the relation(person-party).

Table 4.2: Average accuracy for properties country-currency

Vector Number	Average Accuracy	Percentage
1	0.266666667	26.66666667
2	0.439393939	43.93939394
3	0.509090909	50.90909091
4	0.460606061	46.06060606
5	0.527272727	52.72727273
6	0.545454545	54.54545455
7	0.612121212	61.21212121
8	0.572727273	57.27272727
9	0.6	60
10	0.551515152	55.15151515
11	0.533333333	53.33333333
12	0.590909091	59.09090909
13	0.606060606	60.60606061
14	0.56969697	56.96969697
15	0.581818182	58.18181818
16	0.593939394	59.39393939
17	0.603030303	60.3030303
18	0.621212121	62.12121212
19	0.621212121	62.12121212
20	0.603030303	60.3030303

The system tested 3400 data items against the training dataset for the properties pair person-party. The relation sources are the person and the relation targets are the party name. As the training data were taken randomly, so we got a different number of correct items at every iteration. When the correct items are 1663, the accuracy is:-

$$\text{Accuracy} = (1663/3400) = 0.4891176471$$

For calculating average accuracy we also follow the same instruction from the properties pair capital-country for the properties pair person-party.

For properties person and party, we achieved the highest accuracy for vectors 9 and it is almost 45%. The average accuracy for vector one is 14.885% and the average accuracy is dramatically increased from 14.885% to 33.724% for the vectors two, which is a huge difference. We can confidently say that the goal of the project succeeds.

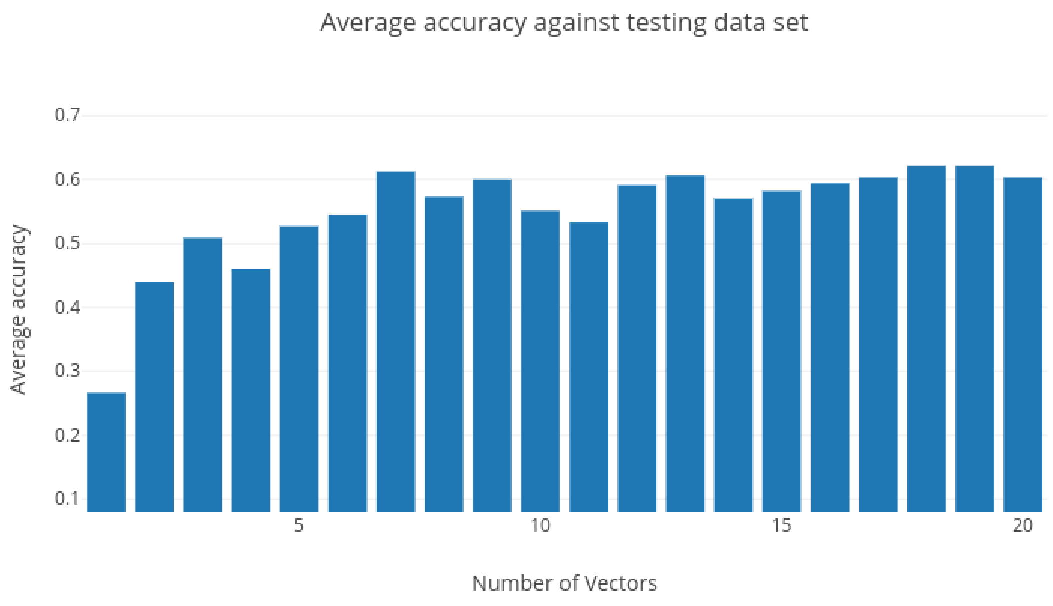


Figure 4.2: Average accuracy against vector number for currency and country

The graphical visualization of the relation(person, party) is shown in the Figure 4.3

Company-Headquarter

For properties company-headquarter we got the following average accuracy for the corresponding number of vectors, shown in Table 4.4.

The lowest average accuracy is received by the properties company-headquarter compared to others for vector one and it is around 2.869%. Even if it is so less for one vector, but it has improved the highest performance with the increasing number of vectors. The average accuracy is dramatically increased for vectors 2 and 3 and they are 10.0820% and 14.83606557% respectively. After that, the average accuracy is gradually increased from vectors 4 to 13. The highest average accuracy is achieved by vectors 13 and it is 28.0328%. Thereafter, it has started to walk toward a lower position.

We followed the same procedure here also to calculate the accuracy and average accuracy for the properties company-headquarter as we followed it for the properties capital-country. We tested 122 data items for the properties company-headquarter.

Table 4.3: Average accuracy for properties person-party

Vector Number	Average Accuracy	Percentage
1	0.148852941	14.88529412
2	0.337235294	33.72352941
3	0.313264706	31.32647059
4	0.358264706	35.82647059
5	0.423647059	42.36470588
6	0.411882353	41.18823529
7	0.416647059	41.66470588
8	0.408264706	40.82647059
9	0.444441176	44.44411765
10	0.342352941	34.23529412
11	0.373441176	37.34411765
12	0.385882353	38.58823529
13	0.421705882	42.17058824
14	0.408558824	40.85588235
15	0.428882353	42.88823529
16	0.427058824	42.70588235
17	0.413558824	41.35588235
18	0.396735294	39.67352941
19	0.418058824	41.80588235
20	0.437294118	43.72941176

The graphical visualization of relation(company, headquarter) is shown in the Figure 4.4

The average accuracy for the four relations are graphically captured on one frame in Figure 4.5. From Figure 4.5, it is visible that the accuracy varies from properties to properties. For properties capital-country, the highest average accuracy is around 94% whereas for the properties country-currency highest average accuracy is roughly 62%. Also, the highest average accuracy for the properties person-party and company-headquarter are respectively about 44% and 28%.

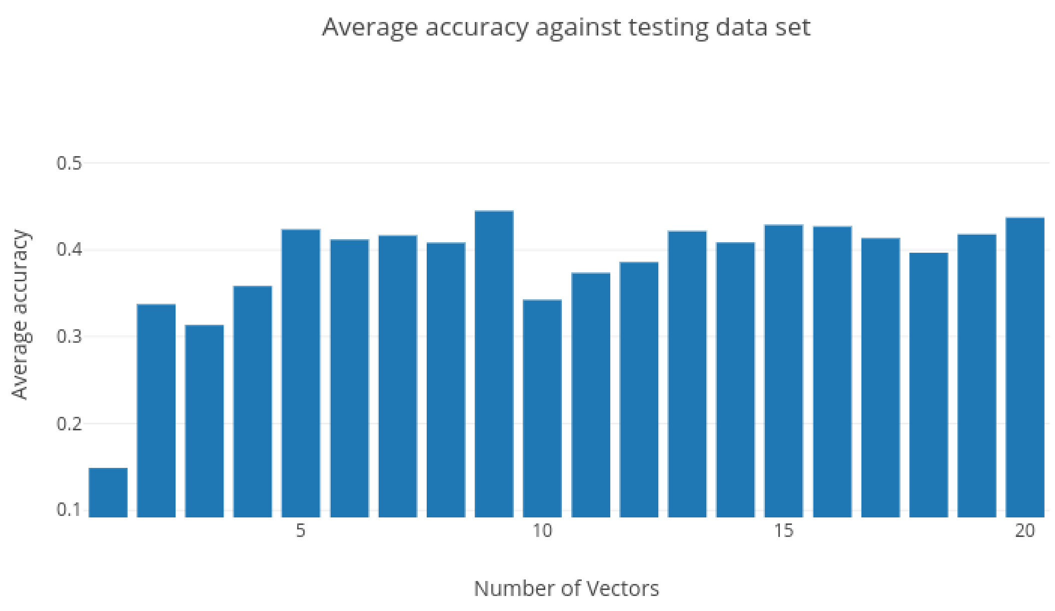


Figure 4.3: Average accuracy against vector number for person and party

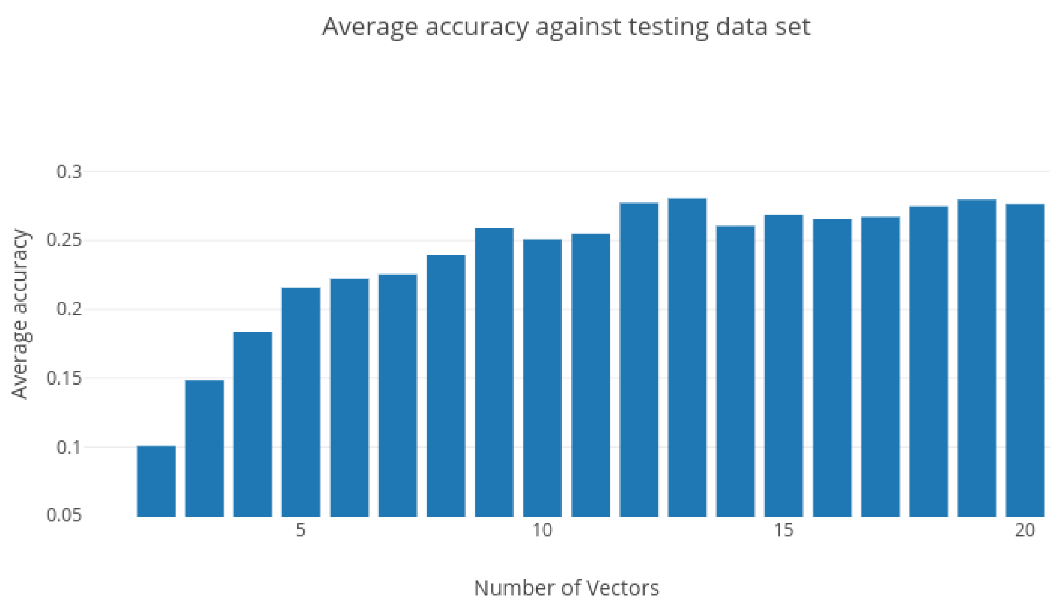


Figure 4.4: Average accuracy against vector number for company and headquarter

Table 4.4: Average accuracy for properties company-headquarter

Vector Number	**Average Accuracy**	**Percentage**
1	0.028688525	2.868852459
2	0.100819672	10.08196721
3	0.148360656	14.83606557
4	0.183606557	18.36065574
5	0.21557377	21.55737705
6	0.222131148	22.21311475
7	0.225409836	22.54098361
8	0.239344262	23.93442623
9	0.259016393	25.90163934
10	0.250819672	25.08196721
11	0.254918033	25.49180328
12	0.27704918	27.70491803
13	0.280327869	28.03278689
14	0.260655738	26.06557377
15	0.268852459	26.8852459
16	0.26557377	26.55737705
17	0.267213115	26.72131148
18	0.274590164	27.45901639
19	0.279508197	27.95081967
20	0.276229508	27.62295082

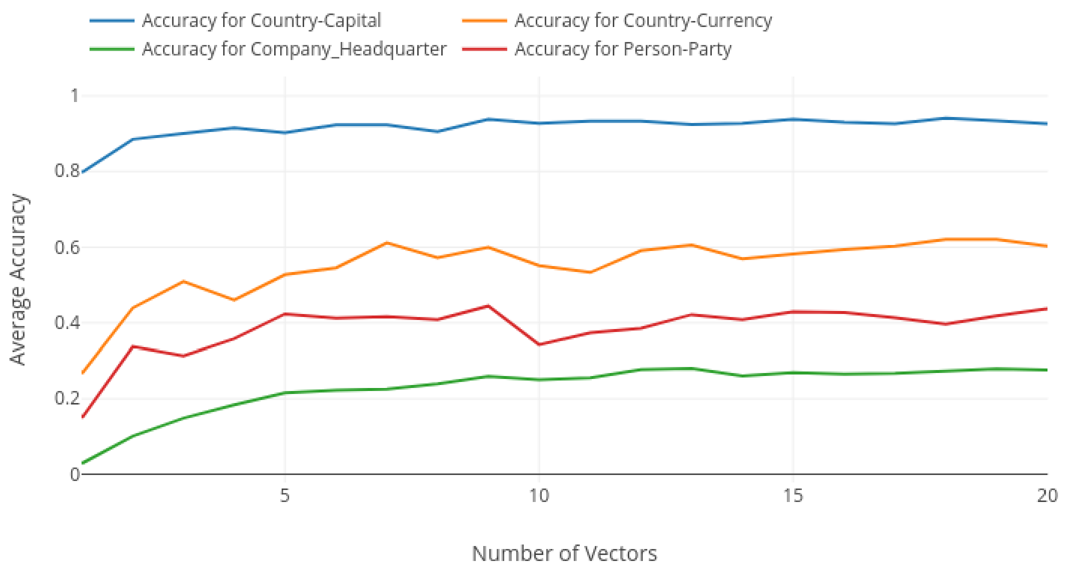


Figure 4.5: Average accuracy against vector number for the four relations

5 Discussion

All of the resources were extracted from DBpedia for the four relations country-capital, currency-country, person-party and company-headquarter by using SPARQL. But it was not possible to think about all of them for further processing. For the above relations, if both the source and target were in the google pre-trained word vector model only then these data were considered. For example in the country-capital relation for the source United_States and target Washington_D.C. from both one was missing. That's why it is not considered for the next step. Since google pre-trained model was used to obtain vector of the word, so if the data is not in the pre-trained model it is not possible to learn vector representation of that word.

As it is mentioned in the result section that accuracy varied from properties to properties. We tried to figure out why the accuracy is less for some properties. The following directions were observed during the evaluation of the system-

1. For properties company-headquarter, in the filtered dataset there was some duplicate data for the company. As an example, Buddle_Findlay appeared three times in the company and Safi_Airways appeared twice. If there were no duplicate we might have obtained better performance.

2. For properties country-currency, some of the currencies were combined with their country name. For instance Iraqi_dinar, Bangladeshi_taka, Ghana_cedi, Hungarian_forint, Malaysian_ringgit etc. The super vector was also constructed with these currency name from the training dataset. Sometimes google's pertained model considers only the currency name without the country. For this reason, the accuracy was less here as compared to capital-country properties.

But for the testing dataset, if the currency is the combination of two words i.e two words are combined with "_" we divided them with "_" and compared the desired results with relation target and also the second part of the divided data. As an example, for the relation target Malaysian_ringgit if the correct answer is Malaysian_ringgit or only ringgit it increments the correct result. Moreover, case-insensitivity is considered here.

Another reason is that, as we discussed in the related work, two-word pairs are relationally similar if their vector differences are similar. In relation(person, party) vector differences are not much similar. That's why the average accuracy is not so high for this properties. The reason is that some relations may be not captured nicely with word embeddings.

In word2vec tool people have been using one-word vector set to predict similarity. We aggregated more than one vector set for better performance. Our highest vector set was twenty. We observed that the performance is really good. The accuracy was increasing with the elevated number of vectors. As an example, for properties company and headquarter, the average accuracy was around 2.9% for one vector and about 28% for the vectors thirteen, which achieved an improvement compared to the analogy task.

6 Conclusion and Outlook

In this paper, we proposed a method to aggregate semantic information stored in word embeddings to predict relation targets of DBpedia properties. The goal of this project is to increase the accuracy of the analogous characteristic of the vectors. The system shows promising results to detect relation target of DBpedia properties with the increasing number of vectors and pointing out for further research.

In this project, super vectors have implemented to increase the accuracy of the analogical reasoning for predicting relation targets of DBpedia properties. This project has some limitations. We have only used the word2Vec tool to get vector representation even though there are some other available techniques such as Glove, dependency-based word embeddings etc. Another observation is that we considered only four SPARQL query against DBpedia to obtain relation (source, target). In future, we have planned to explore more DBpedia data by using SPARQL and will apply other word embeddings methods to predict the relation targets.

Bibliography

Bengio, Yoshua et al. (2003). “A neural probabilistic language model.” In: *Journal of machine learning research* 3.Feb, pp. 1137–1155.

Chen, Dawn, Joshua C Peterson, Thomas L Griffiths (2017). “Evaluating vector-space models of analogy.” In: *arXiv preprint arXiv:1705.04416*.

Chen, Zhiwei et al. (2017). “An exploration of semantic relations in neural word embeddings using extrinsic knowledge.” In: *Bioinformatics and Biomedicine (BIBM), 2017 IEEE International Conference on*. IEEE, pp. 1246–1251.

Gladkova, Anna, Aleksandr Drozd, Satoshi Matsuoka (2016). “Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn’t.” In: *Proceedings of the NAACL Student Research Workshop*, pp. 8–15.

Lassila, Ora, Ralph R Swick, et al. (1998). *Resource description framework (RDF) model and syntax specification*.

Levy, Omer, Yoav Goldberg (2014). “Dependency-based word embeddings.” In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vol. 2, pp. 302–308.

Mikolov, Tomas et al. (2013). “Efficient estimation of word representations in vector space.” In: *arXiv preprint arXiv:1301.3781*.

Mnih, Andriy, Koray Kavukcuoglu (2013). “Learning word embeddings efficiently with noise-contrastive estimation.” In: *Advances in neural information processing systems*, pp. 2265–2273.

Morsey, Mohamed et al. (2012). “Dbpedia and the live extraction of structured data from wikipedia.” In: *Program* 46.2, pp. 157–181.

Teofili, Tommaso (2017). “par2hier: towards vector representations for hierarchical content.” In: *Procedia Computer Science* 108, pp. 2343–2347.