# SREC: Computing Semantic Relatedness based on Entity Comparison in Knowledge Graph*

Jiapeng[1]2

[1] Soochow University, Suzhou JiangSu province, China
[2] trajepl@gmail.com

**Abstract.** In this paper, we propose a model to compute semantic relatedness based on entity comparison in knowledge graph. Before the knowledge graph appeared, the wikipedia was the most frequently used knowledge resource, as it can provide a so broad knowledge coverage that many researches which are based on wikipedia for computing semantic relatedness have obtained more accurate results than those based on WordNet, search engines etc. However the wikipedia-based methods usually require a significant preprocessing. In order to avoid the burden of preprocessing, some researchers consider to utilize knowledge graph to compute semantic relatedness. These researches achieve great results which are on par with wikipedia-based methods. Deficiently, some of information in knowledge graph is missed in these researches. To get an approximation to human relatedness judgement more accurately, we propose a method which utilizes the comparison of entities in knowledge graph. Our model considers more information to estimate semantic relatedness between words. The experiment based on gold dataset shows that our model outperforms the state-of-the-art model.

**Keywords:** Semantic Measure · Semantic Relatedness · Graph Embedding.

## 1 Introduction

Computing semantic relatedness(SR) between two elements(words, sentences, texts etc.) is a fundamental task for many applications in Natural Language Processing(NLP) such as lexicon induction[16], Named Entity Disambiguation[7], Keyword Extraction [27] and Information Retrieval[6]. Besides those NLP applications, in the aspect of opinion spam problem[19] and image classification[9], semantic relatedness measurement plays a great role as well. In this paper we focus on computing semantic relatedness between two words based on entity comparison in knowledge graph.

It has long been thought that when humans measure the relatedness between a pair of words, a deeper reasoning which requires a large amount of knowledge is triggered to compare the concepts behind the words. In the past years, many researchers have worked for semantic relatedness measurement and have made great achievements. From the aspect of data resources which they used, there are: i) *The lexical databases* such as WordNet[15] or Wikithionary[26] play a great part in relatedness measurement. This data resource provides precise lexical information, but misses the semantic information.

* Supported by organization x.

ii) WikiRelated[20], ESA[5], WLM[26] and so on exploit *the large corpora wikipedia* to compute semantic relatedness. Wikipedia-based methods usually outperform the lexical-based methods, because the wikipedia which is built artificially contains abundant semantic information. iii) Recently, there are some researchers having attached importance to measure semantic relatedness in knowledge graph. The SensEmbed [8] leveraged BabelNet[3] to annotate the dump of wikipedia, and exploited word2vec[10] to train the sense-annotated wikipedia to get distributed representation of different word senses. Essentially this method is based on *the large corpora* and needs a significant preprocessing and data transformation efforts. The REWOrD [14] is purely knowledge graph based model which proposed an approach which exploited the graph nature of RDF and SPARQL query language to access knowledge graph. It not only obtained the comparable result with the state-of-art model at that moment, but also avoided the burden of preprocessing and data transformation.

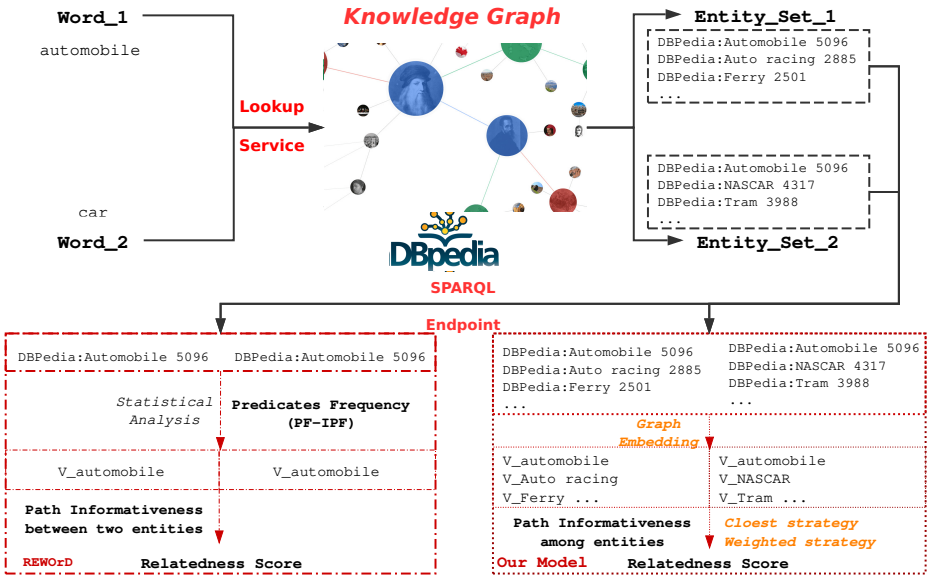



**Fig. 1.** Subgraph in knowledge graph

Though REWOrD[14] achieved great balance between semantic relatedness measurement and data preprocessing. However, they missed some factors which might contribute to semantic relatedness measurement. Firstly given two words as input, the first step is to find corresponding entities in knowledge graph. Obviously, there are usually more than one corresponding entities for a single word. As show in figure 1, for an input word *automobile*, for example, we will get *DBPedia:*[4]*Automobile* and

[3] http://babelnet.org
[4] http://dbpedia.org/resource

*DBPedia:Auto_Racing*, *DBPdedia:Ferry* and so on. From the lower-left corner of figure 1, we can see that the REWOrD lost sight of the informativeness of the other entities, while they just consider the entity with the highest rank. Secondly, REWOrD missed some informativeness of *objects* in *triple(subject, predicate, object)* because their strategy just took the *predicates* into account with statistics of frequencies based on the TFIDF exclusively. This method ignored the information hidden in *objects* in a semantic triple.

To get an approximation to human relatedness judgement more accurately and exploit more knowledge implied in knowledge graph, we propose an improved model based on entity comparison in knowledge graph. As show in figure 1, given two words in input, the first step is to get entities that better associated with the concepts triggered by input words. To this end we rely on lookup services provided by DBPdedia which is used in REWOrD as well. In contrast to REWOrD, we consider multiple entities associated with input words rather than just one with hightest rank. Along with the idea which transforms the entities to vector which can be compared easily, we need to get the vectorization representation of entities. However, traditional statistical analysis methods used in REWOrD can not consider both knowledge hidden in *predicates* and *objects* in semantic triple. To make up this weakness, we use graph embedding to train the dataset which contains subgraphs extracted from knowledge graph. Due to the huge amounts of data in knowledge graph, it is difficult to extract the minimal subgraph which can describe the entities accurately. We do this by building a directed graph which contains all relevant entities and attributes which describe the entities set. After training and computing the distance between vectors, we can get multiple relatedness scores. We use two *weighted* strategy to combine the relatedness scores as the final semantic relatedness score. Experimental results show that our model outperforms the REWOrD and some of other wikipedia-based methods.

In this paper, we propose a threefold model.

i) Given a pair of words, the first job is to query the corresponding entities. In order to use the embedding technique to train the dataset, we construct a graph which contains all related entities, attributes and relations between the corresponding entity pairs.

ii) We exploit embedding technique for knowledge graph instead of Word2vec to train the subgraph extracted from the knowledge graph. Then we can get a distributional representation(vector) for each entity and predicate.

iii) We can get multiple relatedness scores after a full link between these two sets of entities. Inspired by [8], we utilize an approach to combine the relatedness scores as the final semantic relatedness score.

This paper is organized as follows. We give the related work about semantic relatedness measurement in section 2. Then we elaborate the threefold model for computing relatedness scores in section 3. Finally, we display detailed illustrations of experiment results which show that our model outperform the state-of-the-art model.

## 2 Related Work

Semantic measures are mathematical tools used to estimate the intensity of the semantic relationship between units of language, concepts or instances, through a numerical

description. Many traditional studies on semantic relatedness utilize different data resources to compute semantic relatedness. There are

i) *the large corpora*, such as wikipedia. The initial model WikiRelate! [20], firstly retrieved the corresponding Wikipedia articles whose titles contain the words in input and estimated relatedness based on strategies in articles of wikipedia. Explicit Semantic Analysis(ESA)[5] represented the meaning of texts in a high-dimensional space They preprocessed the content of Wikipedia to build an inverted index for each word in texts. Relevance was computed by the TFIDF weighting scheme while relatedness was computed by the cosine of the vectors associated to the texts. WikiRelate! and ESA only leveraged texts in Wikipedia and did not consider links among articles. Another model WLM [12] scrutinized incoming/outgoing links to/from articles instead of exploiting texts in Wikipedia articles. WikiWalk [25] extended the WLM by exploiting not only links that appeared in an article (i.e., a Wikipedia page) but all links, to perform a random walk based on Personalized PageRank.

ii) *the lexical databases*, such as WordNet or Wikithionary. In the wordnet-based methods[15], computed semantic relatedness for automatic speech recognition for meetings. This work did not provide a individual result to reveal the efficiency of semantic relatedness measures. The paper [26] introduced Wikithionary as an emerging lexical semantic resource that could be used as a substitute for expert-made resources in AI applications. They chose(1) a path based approach[17], which can be utilized with any resource containing concepts connected by lexical semantic relations. (2) a concept vector based approach [5]. They generalize this approach to work on each resource which offered a textual representation of a concept.

iii) *the knowledge graph*. Recently, many researchers have used the Knowledge Graph as background knowledge to compute semantic relatedness. In BabelRelate[13], they presented a knowledge-rich approach to compute multilingual semantic relatedness which exploited the joint contribution of different languages. Given a pair of words in two languages, The REWOrd[14] proposed an approach that exploited the graph nature of RDF and SPARQL query Language to access knowledge graph. It not only obtained the comparable result with the state-of-art at that moment, but also avoids the burden of preprocessing and data transformation. However, REWOrd lost sight of the informativeness of the other entities, while they just considered the entity with the highest rank associated with the input words. Secondly, it missed some informativeness of *predicates* as their strategy took the *predicates* into account exclusively based on the TFIDF, which ignored the function of *objects* in a semantic triple.

In summary, in order to improve the performance of DBpedia-based model especially the REWOrd, We consider multiple entities associated with input words rather than just one with hightest rank. Then we utilize the method of embedding to generate high-dimensional vector for corresponding entities. For an input pair of words, we get two sets of corresponding entities in DBpedia ont only the one with highest rank, then get multiple relatedness scores after full links between two sets of entities. In order to better fit the judgement of human, we utilize a combinatorial strategy to combine the relatedness scores of pairwise entities.

# 3 Methodology

**Algorithm 1** SREC

**Input:** a set of words pairs $\{(w_1, w_2)\}$
**Output:** a semantic relatedness score $sr \in [0, 1]$

initialize entites set $E_{w_1}$, $E_{w_2}$ empty
initialize the minimal subgraph $graph\_ent$ that describes entites pairs $< e_1, e_2 >$
initialize the vector set $vector\_ent\_set$ that represents entites set $E_{w_1}$, $E_{w_2}$
initialize the vector set $scores\_set$ that record multiple semantic relatedness sorces
$E_{w_1} \leftarrow query\_lookup(w_1)$; $E_{w_2} \leftarrow query\_lookup(w_2)$
**for** $e_1 \in E_{w_1}, e_2 \in E_{w_2}$ **do**
    $graph\_ent \leftarrow query\_spqrql(e_1, e_2)$ (updating)
    $vector\_ent\_set \leftarrow embedding(graph\_ent)$
    $v_{e_1} \leftarrow find(e_1, vector\_ent\_set)$; $v_{e_2} \leftarrow find(e_2, vector\_ent\_set)$
    $scores\_set.append(e_1, e_2, cos(v_{e_1}, v_{e_2}))$
**end for**
$sr \leftarrow weighted(scores\_set)$
**return** $r$

In order to compute semantic relatedness of a pair of words, we propose a model which is threefold. For a given pair of words, we first query the corresponding entities in knowledge graph. We need to construct a specific graph which contains all related entities and attributes between the corresponding entity pairs. Then we use *Starspace*[23] to train the constructed graph. For each entity and relationship, this method produces a representation of vector. In our method, we use cosine function to compare vectors corresponding to word pairs and get the relatedness scores. Besides, in the query step, we would get several corresponding entities for an input word. Inspired by the SenseEmbed[8], we combine the relatedness scores computed from the multiple pairs of entities as the final measure sorce between two words.

## 3.1 Construct graph

Our aim is to compute the semantic relatedness between a pair of words. The process of relatedness measure needs complete and ample background knowledge which can be gathered from knowledge graph, such as DBPedia[5], YAGO[6] etc. The first problem we face is how to obtain knowledge which is associated with given words from knowledge graph. In our model, we utilize the DBPedia as our knowledge base to gather corresponding entities triggered by the given words. We rely on lookup service[7] that is provided by DBPedia to achieve this object. We use $W_{w_i}$ to denote the set of corresponding entities in knowledge graph when we complete a query by word $w_i$. Then

[5] http://wiki.dbpedia.org/
[6] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago
[7] http://lookup.dbpedia.org/api/search/KeywordSearch

we get $W_{w_i} = \{E_1, ..., E_k\}$ where $E_i$ is the $i_{th}$ query entity associated with the given word. As show in Table 1, for a given word pair *(automobile,car)*, we can get the appropriate entities which are described as URIs in DBPedia. These URIs describe entities so accurate that we can access knowledge graph using powerful query language SPARQL[8] to get everything we want.

**Table 1.** Query Entities (*DBPedia:* equals *http://dbpedia.org/resource/*)

| **Words** | automobile | **Refcount** | car | **Refcount** |
|---|---|---|---|---|
| **Entity** | DBPedia:Automobile | 5096 | DBPedia:Automobile | 5096 |
| **Entity** | DBPedia:Auto_racing | 2885 | DBPedia:NASCAR | 4317 |
| **Entity** | DBPedia:Ferry | 2501 | DBPedia:Tram | 3988 |
| **Entity** | DBPedia:Internal_combustion_engine | 1639 | DBPedia:Auto_racing | 2885 |
| **Entity** | DBPedia:Gasoline | 1486 | DBPedia:Ferry | 2501 |

An entity can be surrounded by entities as well as attributes. There is an example to illustrate this structure in knowledge graph. Suppose that there is a person *A*, his age is *24*. His name is *"John A"*. He has a friend called person *B*. This person *B* is an Entity surrounding the *A*. Then the number *24* and the literal *"John A"* are not entities in knowledge graph, but all are the attributes surrounding this person *A*. We can use the URIs shown in table1 to access knowledge graph with the help of SPARQL. Next we need to construct a subgraph that contains all related entities, attributes and relations between the corresponding entity pairs. Inspired by BabelRelate![13], we propose a modified method to get our semantic subgraph. We get query sets of entities $W_{w_1}$ and $W_{w_2}$ individually by words $w_1$ and $w_2$. Then we select the subgraph which contains all the paths which connect entities in $ENT = W_{w_1} \cup W_{w_1}$. Besides, this subgraph also contains all attributes for each entity in $ENT$. We do this by building a directed graph $G = (V, E)$ which contains all relevant information which describes the entities set.

i) We define the set $V$ in $G$ as $V := ENT$ first. The size of set $V$ is not fixed. It would be extended in the following steps. As for the set $E$, we initialize it as empty, i.e., $E := \emptyset$.

ii)The goal of our method is to get the precise vector representations for entities, that requires more complete information which surround the entities. Accordingly, for a specific entity, we not only need to consider its neighbor entities or attributes, but also need to find all paths which connect the nodes in $V$. It is known to all, the shorter length of paths between two entities, the more relative these entities are. We firstly get the one-hop neighbors for each $v \in V$. Secondly, we adopt Depth-First Search(DFS) to go through the knowledge graph. Every time we find a node $v^{'} \in V$ but $v \neq v^{'}$ along a path$(v, v_1, v_2, ..., v_n, v^{'})$, we add all intermedia nodes and edges in this path to $G$, i.e., $V := V \cup \{v_1, ..., v_n\}$, $E := E \cup \{(v, v_1), ..., (v_n, v^{'})\}$.

[8] https://www.w3.org/TR/sparql11-overview/

iii) Next, we get all relevant attributes which are described as literal, number or something else special symbol in knowledge graph. For one entity $v_i \in V$, we collect all the one-hop attributes $\{a_1, a_2, ..., a_k\}$. Then we have $V := V \cup \{a_1, ..., a_k\}$ and $E := E \cup \{(v_i, a_1), ..., (v_i, a_k)\}$ $(v_i \in V)$.

By this way, we extract a subgraph from DBPedia which consists of the relevant information which describes the pair of entities.

### 3.2 Embedding for Subgraph

The constructed graph is fundamentally a multi-relational graph in which an entity is described by a set of discrete *entities* and *attributes*. Fortunately, there have been an excellent work proposed by Facebook AI Research, *StarSpace*[23]. The model works by embedding those entities comprised of discrete features and comparing them against each other. In this section, we introduce the basic contents of *StarSpace* briefly and elaborate how we utilize this model to get the vector representation of entities and relations. *StarSpace* is available as an open-source project at github[9].

In *StarSpace*, to train our model, we need to compare entities which are described by a set of discrete *entities* and *attributes*. Specially, there is the following loss function in $StarSpace$:

$$\sum_{\substack{(a,b)\in V^+ \\ b^-\in V^-}} L^{batch}(sim(a,b), sim(a,b_1^-), ..., sim(a,b_k^-))$$

In our problem, the input data is a graph of $(h, r, t)$ triples, consisting of a head entity $h$, a relation $r$ and a tail entity $t$. Following the original paper which describes $StarSpace$, there are several explanations for this loss function:

1) The positive entity pair (a,b) comes from the set $V^+$ sampled from constructed graph $G$. In our problem, the input is a group of triples $(h, r, t)$. In order to make our input fit the sample batch $(a, b)$, we need to select uniformly at random to get positive sample $V^+$ in two strategies: (i)$a$ consists of the bag of features $h$ and $r$, while $b$ consists only of $t$; (ii)$a$ consists of $h$, and $b$ consists of $r$ and $t$.

2) Negative entities $b^-$ are sampled from the set of possible entities $V^-$. StarSpace utilizes a $k$-negative sampling strategy[10] to select $k$ negative pairs for each batch update. They select randomly whithin the set of entities that can appear in the second argument of the similarity function.

3) The selection of function $sim(.,.)$ is designed as a hyper-parameter: cosine similarity and inner product. In our problem, we adopt cosine similarity for the model as the cosine works better than inner product for larger dataset. This situation is mentioned in the paper of StarSpace.

4) The loss function $L_{batch}$ compares the positive pairs $(a, b)$ with the negative pairs $(a, b_i^-)$, $i = 1, ..., k$. It is also optional between margin ranking loss and negative log loss of softmax. All experiments in $StarSpace$ show the former performed on par or better. Thus we use margin ranking loss as our loss function for computing semantic relatedness.

[9] https://github.com/facebookresearch/StarSpace

5) The method optimization inherit the strategy of stochastic gradient descent(SGD) used in $Starspace$. Each SGD step is one sampling from $V^+$ in the outer sum.

As a result, we take the constructed graph $G$ of $(h, r, t)$ triples as inputs for the training model. For each entity and relation in graph $G$, there is a fixed-length vector which can then be used to compute semantic relatedness via cosine function.

### 3.3 Semantic Relatedness Measure

For a given pair of words($w_m$, $w_n$), we get ($W_{w_m}$, $W_{w_n}$), and $W_{w_m} = \{E_m^1, E_m^2..., E_m^k\}$, $W_{w_n} = \{E_n^1, E_n^2, ..., E_n^k\}$. Then for each entity $E_m^i$ in $W_{w_m}$, we will get learnt embedding vector $\overrightarrow{E}_m^i$. Note that, there might be different number of entities associated with the given word. We just consider the top-$k$ entities in each entities set. An analysis of the impact of $k$ is given in section of experiments.

For a pair ($w_1$, $w_2$), we compute semantic relatedness between their corresponding entities. After that, we get $k * k$ relatedness results where $k$ is number of entities queried from knowledge graph. The task which combines multiple semantic measurement is similar with the work in SenseEmbed[8]. The author captureed the different meanings of a word and transformed word embedding to the sense level. They utilized the weighted combination of comparison in sense level which achieved a high correlation coefficient. In our work, the multiple-pair entities produce different semantic measurement. Traditional work only considers the the relatedness measurement of closest objects. If we only utilize closest combination, the contributions of the other entities would be ignored. Following the work in SenseEmbed, we combine all those relatedness results reasonably to get more human-like measurement.

1) For two entities $E_1$ and $E_2$, we utilize cosine function to compute the distance between two embedding vectors($\overrightarrow{E}_1$, $\overrightarrow{E}_2$):

$$Cos(\overrightarrow{E}_1, \overrightarrow{E}_2) = \frac{\overrightarrow{E}_1 \cdot \overrightarrow{E}_2}{\left\|\overrightarrow{E}_1\right\| \left\|\overrightarrow{E}_2\right\|}$$

2) Following SenseEmbed, we take two strategies to compute semantic relatedness between given words $w_1$ and $w_2$ for comparison. One is conventional approach [3] which considers just the closest entities among multiple vector pairs. There are $W_{w_1}$ and $W_{w_2}$ represent two entities sets associated with two input words $w_1$ and $w_2$. $\overrightarrow{E}_i$ is the learnt embedding vector of entity $E_i$. We can get the formalization for this strategy.

$$Rel_{closest}(w_1, w_2) = \max_{\substack{E_1 \in W_{w_1} \\ E_2 \in W_{w_1}}} Cos(\overrightarrow{E}_1, \overrightarrow{E}_2)$$

However this relatedness measurement approach misses contributions of the other entities. In fact, psychological studies suggest that humans, while comparing similarity between a pair of words, consider different meanings of two words but not only the closest pairs[22]. This claim can be directly popularized to compute relatedness between words. There usually are various meanings for a single word. Analogously we can query multiple entities associated with the word from knowledge graph. Only consider the entities pair which have the closest distance would not be conforming with the way of human thinking.

There is another strategy for computing semantic relatedness, called *weighted*, in which we consider the contributions of different entities associated with given words. The contributions are scaled according to how they relative to the words. Fortunately, the lookup services of DBPedia can return a list of ranked DBpedia resources for a search string or a single word. There is a specific label called $Refcount$ that count the number of Wikipedia page inlinks for a resource. This number is required for ranking. We can utilize this number to estimate the dominance of each specific entity resource in knowledge graph. To this end, we use an operation of normalization. For each entity $E \in W_{w_i}$, we get the dominance of $E$ by dividing the value of $Refcount$ by the sum of all $Refcount$ value of entities in $W_{w_i}$:

$$d(E) = \frac{refcount(E)}{\sum_{E' \in W_i} refcount(E')}$$

Besides, those entities which are closer would play a more important role in computing semantic relatedness. Following the SenseEmbed, we model this by biasing the relatedness computation towards closer entities through a power function with parameter $\alpha$. The relatedness of a pair of words $w_1$ and $w_2$ is computed by using $weighted$ strategy:

$$Rel_{weight}(w_1, w_2) = \sum_{E_1 \in W_1} \sum_{E_2 \in W_2} d(E_1) d(E_2) Cos(\overrightarrow{E}_1, \overrightarrow{E}_2)^{\alpha}$$

In this strategy, we given the entity pairs which is closer a more important role to determine the final semantic relatedness score. Experiments in below show that the $weighted$ strategy outperforms the $closest$.

## 4 Experiment

In this section, we conduct extensive experiments on different datasets which contain the semantic measurement by human perceptions. We compute the Spearman correlation coefficient between results of experiments and scores of human judgement to evaluate the performance of our model. The model is implemented on cpu Core-7-7700K@4.20GHz×8 machine with 16GB memory and ArchLinux platform.

### 4.1 Dataset

To evaluate models for semantic relatedness measurement, a common approach is to compare the scores from the model and the scores provided by humans performing the same task. This approach provides a model-independent way for evaluating measures of relatedness. A good number of datasets record the scores of human quantitative judgement for semantic relatedness such as *MC-30*[11], *RG-65*[18], *REL-122*[21], *MEN*[2], *YP-130*[24], *WS*[1], *wordsim-353*[4] and so on. Some of these datasets are established

with the measurement of similarity called *similarity dataset*. Some others are *relatedness dataset*. Note that all these datasets are English-language words.

1)*similarity dataset*: *MC-30*, *RG-65*, *YP-130* and *wordsim-353* are all established for computing similarity. *RG-65* is the classical similarity dataset which contains 65 pairs of words. *MC-30* contains 30 pairs of words and is the subset of *RG-65*. *YP-130* is established for computing verb similarity which contains 130 pairs of verb. *wordsim353* contains two parts that are annotated by different groups of annotators. The first set (set1) contains 153 word pairs along with their similarity scores assigned by 13 subjects. The second set (set2) contains 200 word pairs, with their similarity assessed by 16 subjects. All these above are mere similarity datasets which just consider a particular case of relatedness. There is another hybrid dataset *WS* contains two sets of English words pairs along with human-assigned similarity and relatedness judgements, called *WS-sim* and *WS-Rel* separately. *WS-sim* contains 203 pairs of words along with similarity judgement, and *WS-rel* contains 252 along with relatedness judgement.

2)*relatedness dataset*: Compared to the similarity datasets, there are a small number of relatedness datasets such as *WS-Rel*, *MEN* and *REL-122*. *WS-Rel* contains 252 pairs of words with human-assigned relatedness judgements. *MEN* contains 3000 pairs of words which do not instruct the subjects about the difference between similarity and relatedness. Due to the great number of human-assigned relatedness judgements, *MEN* can be used to train and test computer algorithms implementing semantic similarity or relatedness measures. We would not consider this collection in our experiments because of the illegibility of semantic measurement. Another dataset *REL-122* is a new relatedness norm, and contains a set of human-assigned relatedness scores for 122 pairs of nouns.

The difference between similarity dataset and relatedness dataset is how human assign the score for a given pair of words. A common example is the pair of words *"wheels-car"*. The *wheels* is more related with the *car*, but they are dissimilar. The paper [21] created two additional experimental conditions in which subjects evaluated the relatedness of noun pairs from the similarity dataset *MC-30*. We select ten pairs of words shown in table 2.

**Table 2.** Relatedness vs MC-30 similarity

| # | Noun Pairs | | Sim. | Rel. |
|---|---|---|---|---|
| **1.** | car | automobile | 3.92 | 4.00 |
| **2.** | gem | jewel | 3.84 | 3.98 |
| **3.** | coast | shore | 3.70 | 3.97 |
| **4.** | journey | car | 1.16 | 3.00 |
| **5.** | forest | graveyard | 0.84 | 2.01 |
| **6.** | coast | hill | 0.87 | 1.59 |
| **7.** | shore | woodland | 0.63 | 1.63 |
| **8.** | lad | wizard | 0.42 | 2.12 |
| **9.** | crane | implement | 1.68 | 0.90 |
| **10.** | noon | string | 0.08 | 0.14 |

We can see that the relatedness for pairs that are related but dissimilar(e.g., *journey-car* and *forest-graveyard*). This indicates that asking subjects to evaluate similarity instead of relatedness can significantly impact the results in studies of semantic relatedness measurement. However, previous researchers do not consider the semantic difference among some datasets. For example, in [5], [25], [14] and so on, the researchers all regard the dataset *MC-30* as relatedness dataset to conduct experiments. Following these reasearches, the similarity dataset *MC-30* and *RG-65* would be leveraged in our experiments besides the relatedness datasets *rel-122* and *WS-rel*. Moreover, we extract the relatedness scores from paper [21]. And some of these pairs are shown in table 2 for dataset *MC-30*.

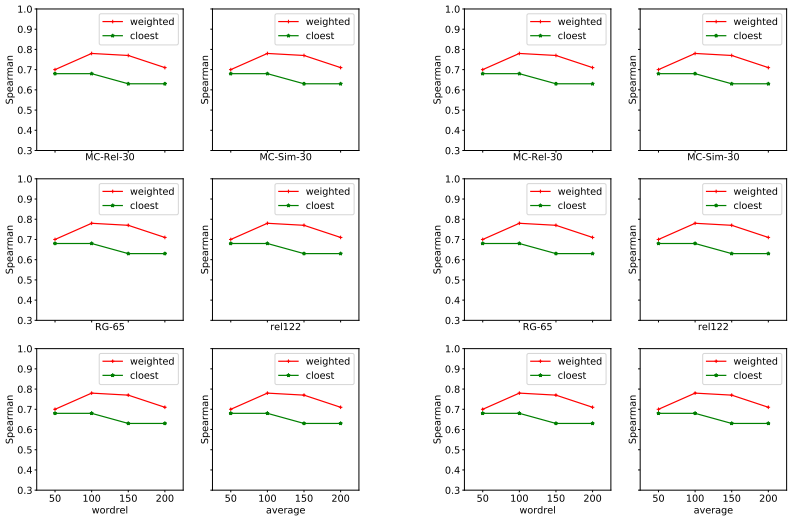
For a given pair of words, we firstly get two sets of corresponding entities in DBPedia. Then we adopt a method of embedding to train the subgraph which is extracted from DBPedia based on queried entities. Finally, we get multiple relatedness scores after a full link between two sets of entities. In order to better fit the judgement of human, we utilize a weighted strategy to combine multiple relatedness scores of pairwise entities.

### 4.2 Parameter tuning

We can recall from section 3.2 and 3.3 that there are some parameters which may have an impact on semantic relatedness measurement. The final relatedness socre would be affected by the dimension of vector as well as the number of entities associated with given words. Besides, we tune the $\alpha$ parameter for the *weighted* strategy on *WS-rel* dataset. We pick *WS-rel* since there are not many comparison systems in literature that report results on this dataset. Another reason is that the quantity of *WS-rel* is greater than other datasets. It is comprehensive for our parameter tuning using dataset *WS-rel*. Finally, we find the optimal values for $\alpha$ to be 7.

As you can see by the line chart 2(a), we exhibit the impact of dimensions of learnt vector for relatedness measurement with *closest* and *weighted* strategy. We conduct results based on datasets which consist of *MC-Rel-30*, *MC-Sim-30*, *RG-65*, *rel-122*, and *WS-rel*, and we get the average values of the results in five datasets shown in the sixth line chart finally. It is obvious that the we get the optimal relatedness scores when the dimension of vector is assigned to *100*.

In the figure 2(b), we draw the variation trend of semantic relatedness scores following the increase of the quantity of entities queried by given words. The slope of correlation coefficient curve rises gently with the increase of number of queried entities. Distinguishingly, when the quantity of queried entities is greater than 5, the correlation coefficient curve has a concave shown in pic *X* because extra and overmuch entities would bring noise for the final semantic relatedness scores.




(a) The impact of Dimension of vector for relatedness measure

(b) The impact of the number of entities queried by given words

**Fig. 2.** Parameter Tuning

**Table 3.** Spearman correlation performance on five word similarity and relatedness datasets

| Measure | Dataset | | | | | Averrage |
|---|---|---|---|---|---|---|
| | MC-rel-30 | MC-30 | RG-65 | rel122 | wordrel | |
| **WikiRelate!** | – | 0.45 | 0.52 | – | – | |
| **ESA** | – | 0.75 | 0.82 | – | – | |
| **WLM** | – | 0.70 | 0.64 | – | – | |
| **WikiWalk** | – | 0.61 | – | – | – | |
| **REWOrD** | – | 0.72 | 0.78 | – | – | |
| **Pando-closest** | | 0.81 | | | | |
| **Pando-weighted** | | **0.85** | | | | |

Compared to previous method of semantic relatedness measurement, we run our model on five dataset *MC-Rel-30*, *MC-Sim-30*, *RG-65*, *rel-122*, and *WS-rel*, and report the *Spearman* correlation coefficient performance for the two strategies of our model. It can be seen that our model proves to be highly reliable on semantic relatedness measurement tasks. Our model obtains the best performance on most datasets. In addition, our approach in dataset *MC-Rel-30* outperforms the results in dataset *MC-Sim-30* which proves that our model is more suitable for computing semantic relatedness than similarity.

## 5 Conclusion

In this work, we focus on computing semantic relatedness to get an approximation to human judgement. We utilize the Knowledge Graph DBPedia which is derived from wikipedia as background knowledge. For an input pairwise word, we get two sets of corresponding entities from DBpedia. Then we propose a model to extract a subgraph which contains complete information surrounding two sets of entities. We use a model of knowledge graph embedding to train the subgraph and generate high-dimensional vector for each entity and relation. Finally, we get multiple relatedness scores after a full link between two sets of entities. In order to better fit the judgement of human, we utilize a weighted strategy to combine multiple relatedness scores. The experiments based on golden dataset show that our model outperforms the state-of-the-art model in semantic relatedness measurement.

## References

1. Agirre, E., Alfonseca, E., Hall, K.B., Kravalova, J., Pasca, M., Soroa, A.: A study on similarity and relatedness using distributional and wordnet-based approaches. In: Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA. pp. 19–27 (2009)
2. Bruni, E., Tran, N., Baroni, M.: Multimodal distributional semantics. J. Artif. Intell. Res. **49**, 1–47 (2014)

3. Budanitsky, A., Hirst, G.: Evaluating wordnet-based measures of lexical semantic relatedness. Computational Linguistics **32**(1), 13–47 (2006)
4. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: the concept revisited. ACM Trans. Inf. Syst. **20**(1), 116–131 (2002)
5. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007. pp. 1606–1611 (2007)
6. Gurevych, I., Müller, C., Zesch, T.: What to be? - electronic career guidance based on semantic relatedness. In: ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic (2007)
7. Han, X., Zhao, J.: Structural semantic relatedness: A knowledge-based method to named entity disambiguation. In: ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden. pp. 50–59 (2010)
8. Iacobacci, I., Pilehvar, M.T., Navigli, R.: Sensembed: Learning sense embeddings for word and relational similarity. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers. pp. 95–105 (2015)
9. Leong, C.W., Mihalcea, R.: Measuring the semantic relatedness between words and images. In: Proceedings of the Ninth International Conference on Computational Semantics, IWCS 2011, January 12-14, 2011, Oxford, UK (2011)
10. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013)
11. Miller, G.A., Charles, W.G.: Contextual correlates of semantic similarity. Language and Cognitive Processes. **6**(1), 1–28 (1991)
12. Milne, D., Witten, I.H.: An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In: In Proceedings of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy. pp. 25–30 (2008)
13. Navigli, R., Ponzetto, S.P.: Babelrelate! A joint multilingual approach to computing semantic relatedness. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada. (2012)
14. Pirrò, G.: Reword: Semantic relatedness in the web of data. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada. (2012)
15. Pucher, M.: Wordnet-based semantic relatedness measures in automatic speech recognition for meetings. In: ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic (2007)
16. Qadir, A., Mendes, P.N., Gruhl, D., Lewis, N.: Semantic lexicon induction from twitter with pattern relatedness and flexible term length. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. pp. 2432–2439 (2015)
17. Rada, R., Mili, H., Bicknell, E., Blettner, M.: Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man, and Cybernetics **19**(1), 17–30 (Jan 1989)
18. Rubenstein, H., Goodenough, J.B.: Contextual correlates of synonymy. Commun. ACM **8**(10), 627–633 (1965)
19. Sandulescu, V., Ester, M.: Detecting singleton review spammers using semantic similarity. In: Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume. pp. 971–976 (2015)
20. Strube, M., Ponzetto, S.P.: Wikirelate! computing semantic relatedness using wikipedia. In: Proceedings, The Twenty-First National Conference on Artificial Intelligence and the

Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA. pp. 1419–1424 (2006)
21. Szumlanski, S.R., Gomez, F., Sims, V.K.: A new set of norms for semantic relatedness measures. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers. pp. 890–895 (2013)
22. Tversky, A., Gati, I.: Features of similarity. Psychological Review **84**, 327–352 (1977)
23. Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., Weston, J.: Starspace: Embed all the things! CoRR **abs/1709.03856** (2017)
24. Yang, D., Powers, D.M.W.: Verb similarity on the taxonomy of wordnet. In: In the 3rd International WordNet Conference (GWC-06), Jeju Island, Korea (2006)
25. Yeh, E., Ramage, D., Manning, C.D., Agirre, E., Soroa, A.: Wikiwalk: Random walks on wikipedia for semantic relatedness. In: Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing, August 7, 2009, Singapore. pp. 41–49 (2009)
26. Zesch, T., Müller, C., Gurevych, I.: Using wiktionary for computing semantic relatedness. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008. pp. 861–866 (2008)
27. Zhang, W., Feng, W., Wang, J.: Integrating semantic relatedness and words' intrinsic features for keyword extraction. In: IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013. pp. 2225–2231 (2013)