Assessing semantic similarity between concepts using Wikipedia based on nonlinear fitting

Guangjian Huang[1], Yuncheng Jiang[1⋆], Wenjun Ma[1], and Weiru Liu[2]

[1] School of Computer Science, South China Normal University, Guangzhou, China
[2] School of Computer Science, Electrical and Electronic Engineering, and Engineering Maths, University of Bristol, UK.

Abstract. Feature-based methods of semantic similarity with Wikipedia achieve fruitful performances on measuring the "likeness" between objects in many research fields. However, since Wikipedia is created and edited by volunteers around the world, the preciseness of these methods more or less are influenced by the incompleteness, invalidity and inconsistency of the knowledge in Wikipedia. Unfortunately, this problem has not got enough attention in the existing work. To address this issue, this paper proposes a novel feature-based method for semantic similarity, which has three parts: low frequency features removal, the similarities of generalized synonyms computing, and weighted feature-based methods based on nonlinear fitting. Moreover, we show that our new method can always get a better Pearson correlation coefficient on one or more benchmarks through a set of experimental evaluations.

Keywords: Semantic similarity · Wikipedia · Nonlinear fitting

1 Introduction

Semantic similarity computation can estimate the "likeness" (similar) between objects (words, concepts or sentences) based on background knowledge and contextual awareness. Thus, it plays an essential role in many research fields, such as information retrieval [3] and natural language processing [11]. Despite its usefulness, robust measurement of semantic similarity for large scale natural language processing application remains a challenging task. Many works have been developed in the last few years, especially with the increase in feature-based methods with Wikipedia [2, 5, 6, 11, 14]. For these methods, Wikipedia serves as a huge size knowledge resource with significant coverage and feature-based measures can exploit more semantic knowledge than edge-counting approaches with the evaluations about the commonalities and differences of compared concepts. Thus, such methods have the potential to solve the task.

However, since Wikipedia is a free encyclopedia that anyone can edit, it is too hard to be sure about the quality of the knowledge in Wikipedia. As a result, by relying on the features extracted from Wikipedia, this kind of methods

⋆ Corresponding author (ycjiang@scnu.edu.cn).

suffer from a critical drawback: the precision of the semantic similarity results is deeply influenced by the incompleteness, invalidity and inconsistency of the knowledge in Wikipedia. Unfortunately, in the existing researches, this issue has been ignored. Another problem of feature-based approaches is their dependency on the weighting parameters that balance the contribution of each feature : (a) in the methods that without weights [6], features are treated equally. Thus, the important underlying statistics of knowledge resource is ignored. (b) In the weighted methods, it is a complex task to assign weights for each feature with Wikipedia. In the literature[5, 6, 11], researches usually use some small subsets of whole Wikipedia (*i.e.*, Wikipedia category graph) for feature weights assignment. But such subsets do not offer as much coverage as Wikipedia.

To address the above issues, in this paper, based on [6], we propose some new weighted feature-based methods by the nonlinear fitting. So we can make sure different features make different contributions to semantic similarity computation. Specifically, we first remove low-frequency features to reduce noise. Then we provide a new way to compute the similarities of generalized synonyms, based on the number of their categories. After that, we use nonlinear fitting to construct some weighted feature-based methods for semantic similarity. Finally, we show that our new methods can always get a better Pearson correlation coefficient on some benchmarks through a set of experimental evaluations. We will test our methods in the following benchmarks, RG65 [13], MC30 [9], 353tc [1], Sim666 [4], Jiang30 [6].

This paper advances the state-of-the-art on the topic of semantic similarity computation in the following aspects: (i) we propose a set of methods based on nonlinear fitting to increase the precision of semantic similarity results based on Wikipedia. (ii) Our new methods can reduce the influence of the incompleteness, invalidity and inconsistency of the knowledge in Wikipedia on the semantic similarity computation results. (iii) We give a new features weights assignment method for semantic similarity by the nonlinear fitting. (iv) Several widely used benchmarks have been considered to enable an objective comparison.

The rest of the paper is organized as follows. Section 2 discusses some background knowledge and related work. Section 3 analyzes some limitations of previous researches and talks about how to improve those limitations step by step. Then we evaluate our methods on benchmarks in the next section. Finally, Section 5 concludes the paper with future work.

2 Background and related work

This section provides a brief introduction about Wikipedia and discusses related work about feature-based methods of semantic similarity between concepts.

Wikipedia is a free, multilingual, largest, most widely used and up to date encyclopedia in existence. English version of Wikipedia contains over 5 million articles. A Wikipedia article offers a great deal of textual information and features (such as redirects, glosses, hyperlinks, disambiguation pages and categories). An article is the basic unit of Wikipedia, which contains text about

a specific concept. Each article is assigned a *title* which is also referred as a *concept*. The opening paragraph of an article provides its summary, referred as *gloss*. An article has a set of anchors, which are internal hyperlinks to other related Wikipedia articles and categories are used to group related articles. Some synonyms share the same article and polysemic words refer to many articles. Our experiments are based on the Wikipedia dump of 05 January 2019.

The basic idea of feature-based methods based on Wikipedia is that concepts with more shared features are more similar than the concepts with less shared features [7]. Thus, the similarity of compared concepts can be obtained by their features comparison. To this end, in [6, 5, 11], they propose a series of feature-based methods that concepts are defined by four features (*i.e.*, anchor, category, gloss, and synonym) extracted from the concepts' articles in Wikipedia. In their definitions, a concept is denoted as Con, and $Con = (Anchor(A), Category(C), Gloss(G), Synonym(S))$, where $A = \{a_1, \cdots a_n\}$ is the set of all the internal hyperlinks to other Wikipedia concepts in the Wikipedia article of Con and $C = \{c_1, \cdots c_k\}$ is set of categories that Con belongs to, $G = \{t_1, \cdots t_i\}$ is the set of all the terms that extracted from the first paragraph of the article of Con and $S = \{s_1, \cdots s_m\}$ is a set of alternative aliases (Redirects) or synonyms of Con. Thus, the similarity between two concepts Con_1 and Con_2 is defined as:

$$Sim(Con_1, Con_2) = S_{con}(\Im(A), \Im(C), \Im(G), \Im(S)) \tag{1}$$

where $\Im(K)$, $K \in \{A, C, G, S\}$ represents the similarity of four feature sets of anchors, categories, glosses, and synonyms respectively. $K \in \{A, C, G, S\}$ is a pair of feature sets for the compared concepts. For example, $\Im(A) = \Im(A_1, A_2)$ and A_1 (or A_2) represents the sets of anchors of $Con1$ (or $Con2$) respectively. Hence, function $\Im(K) = \Im(x, y)$ is the similarity of set x and set y. Specifically, $\Im(x, y)$ can be defined based on X-similarity [10] or RE-approach [12], i.e.,

$$\Im_X(x, y) = \frac{|x \cap y|}{|x \cup y|}, \tag{2}$$

$$\Im_{RE}(x, y) = \frac{|x \cap y|}{|x \cap y| + \alpha|x \setminus y| + (1 - \alpha)|y \setminus x|}. \tag{3}$$

Finally, function S_{con} can be considered as an aggregation operator to combine the similarity of four feature sets of anchors, glosses, categories and synonyms. Thus, it can be mean function or max function, and so on.

Based on Equation (1), Jiang *et al.* have chosen different forms for function S_{con} and $\Im_X$ (or $\Im_{RE}$) in Equations (2) and (3) to get different methods of semantic similarity in [6]. After comparing the experimental results of twelve different methods, it is concluded that the following four methods will get a better results in different benchmarks than other methods:

$$SimFir(Con_1, Con_2) = (\Im_X(A) + \Im_X(C) + \Im_X(G) + \Im_X(S)) \times 0.25, \tag{4}$$

$$SimSec(Con_1, Con_2) = (\Im_{RE}(A) + \Im_{RE}(C) + \Im_{RE}(G) + \Im_{RE}(S)) \times 0.25, \tag{5}$$

$$SimThi(Con_1, Con_2) = \max(\Im_X(A), \Im_X(C), \Im_X(G), \Im_X(S)), \quad (6)$$
$$SimFou(Con_1, Con_2) = \max(\Im_{RE}(A), \Im_{RE}(C), \Im_{RE}(G), \Im_{RE}(S)), \quad (7)$$

where function $\Im_X(x, y)$ is defined as Equation (2) and $\Im_{RE}(x, y)$ is defined as Equation (3). In $\Im_{RE}(x, y)$, α is defined as 0.5 in default in [6]. In this paper, we will show how to improve the above four methods step by step.

3 Weighted feature-based methods with nonlinear fitting

Since Wikipedia is a free encyclopedia that anyone can edit, the contents of an article are influenced by the knowledge and culture of the people who edit this article. It will lead to the following issues. Firstly, incompleteness, some features of a concept are not included in the contents. Secondly, inconsistency, a feature is supposed to be a common feature between two concepts, but in fact, it belongs to only one concept. Finally, invalidity: a feature is not the right or appropriate feature for a concept. For instance, both "Potato" and "Tomato" are not in the category "Vegetables", but "Pomato" is. There are many terms extracted from glosses just appeared in only one article, such as "medula", "mmeli", "treatmentknown", "treatmentth" and "absolutein".

To reduce the influence of such issues for the semantic similarity computation result, in this section, based on the four methods in Equations (4)-(7), we propose our new weighted feature-based method with nonlinear fitting. These methods include two stages: first, we remove the low-frequency features to reduce noise (i.e., reduce the influence of invalidity) and the similarities computation of generalized synonyms (i.e., reduce the influence of inconsistency). Second, we apply nonlinear fitting for the weights assignment of four feature-based methods in Equations (4)-(7). In the second stage, to mitigate incompleteness and invalidity, we will try to use weights in [-1,1], and the sum of all features' weights is not limited to 1. Here, the negative weights mean that in case some invalidity features are considered, we use negative weights to reduce the influence of such invalidity to the final similarity between two concepts.

Finally, to make our methods more realistic, we will explain our methods step by step in the following benchmarks, **RG65** [13], **MC30** [9], **353tc** [1], **Sim666** [4], **Jiang30** [6].

3.1 Low-frequency features and synonym

To construct a feature-based method based on Wikipedia, we first need to extract Wikipedia features for the concepts in different benchmarks. In this paper, we use JWPL[3](Java Wikipedia Library) to extract Wikipedia features. We also use MySQL to manage data. For the gloss of every concept, we remove the stop words, special characters, punctuation and numbers. This preprocessing is the same as that in [6].

[3] https://dkpro.github.io/dkpro-jwpl/

Therefore, by 5 million articles in Wikipedia, for all the concepts in the benchmarks mentioned above (*i.e.*, **RG65**, **MC30**, **353tc**, **Sim666**, **Jiang30**), we can extract 12,028 unique words in glosses, 104,963 unique anchors. However, we find that 513 words just appear in ten or less Wikipedia glosses and 1,802 anchors just appear in ten or less Wikipedia articles and. Since there are 5 million articles in Wikipedia and such anchors or words only appeared in a few articles, we call them as the low-frequency features. There is a high possibility that such low-frequency features are coming from mistakes. For instance, there are many terms extracted from gloss just appeared in only one article, such as "medula", "treatmentth" and "absolutein". As a result, such low-frequency features can be considered as a noise which reduces the similarities of concepts.

To address this issue to improve the precision of semantic similarity computation result, in our new method, we first remove the low-frequency features as follows: (a) we will remove the anchors and words that appear no more than 200 times.[4] (b) For categories, we only remove the hidden categories.[5] (c) We do not remove the low-frequency categories since it is reasonable for some categories just appear in only one article.

Hence, many concepts in Wikipedia are ambiguous and will be redirected to the same article. If we compute their similarity by the existing feature-based methods [2, 5, 6, 11, 14], their similarity will be the maximum value 1. Besides, the same will happen to polysemic words that have common senses as well. For instance, in [14], they define the similarity of two words (w_1, w_2) as follows.

$$sim(w_1, w_2) = max_{c_1 \in s(w_1), c_2 \in s(w_2)}(sim(c_1, c_2)), \tag{8}$$

where $s(w)$ denotes a set of concepts that are senses of word w. Therefore, they will get $sim(Football, Soccer) = 1$ in the scope of $[0, 1]$. However, "soccer" means American football and for the rest of the world, "football" means association football. Thus, it is more reasonable to assign $sim(football, soccer) < 1$.

In this paper, both redirected pages and polysemic words that have common senses are called generalized synonyms (synonyms for short). When two synonyms represent different categories of objects, their senses in context will be different. Formally, when Con_1 and Con_2 are synonyms, we have

$$Sim(Con_1, Con_2) = F(B, C) = \begin{cases} B + \frac{R}{n(C)}, & \texttt{if } n(C) > 0, \\ B, & \texttt{if } n(C) = 0, \end{cases} \tag{9}$$

where $n(C)$ represents the number of categories of Con_1 and Con_2. Coefficient "B" ($B \in [0.5, 1]$) is a constant that measures the similarity for every synonym pair and constant "R" measures the degree of influence that the number of categories can have on similarity. Since the influence of the number of categories

[4] In the experiments, we have tried different thresholds for low-frequency features between 0 and 2000, and the number 200 works better than others.

[5] Hidden categories are used for maintenance of the Wikipedia project which is not part of the encyclopedia. For instance, "1911 Britannica articles needing updates from January 2016" is a hidden category.

on similarity can be either positive or negative, we have $R \in [-1, 1]$. Notice that synonyms share the same article in Wikipedia, so Con_1 and Con_2 have the same number of categories.

The intuition of Equation (9) is that for the compared concepts which are synonyms, we first give them an initial similarity. Then the more categories they have, the less deviation of the initial similarity they will be. Hence, when R is positive, B is the lower bound of the similarities. Thus, the synonym pairs that have less common categories will be more similar. On the contrary, negative R means B is the upper bound of the similarities. Thus, the synonym pairs that have more common categories will be more similar.

Now, we considering how to give appropriate values for constants B and R in Equation (9). First, we give a formal definition of discrete spaces as follows:

Definition 1 (Discrete space). *A one dimension discrete space* $D = \{a, a + d, a + 2d, a + 3d \cdots a + (n-1)d, a + nd, b\}$ *is defined as* $D = [a, b, d]$. *Define* n *dimension discrete space as* $D^n = \{(x_1, x_2 \cdots x_n) | x_i \in D, i = 1, 2 \cdots n\}$.

Then, considering different values of B in discrete space $[0.5, 1, 0.05]$ and R in discrete space $[-1, 1, 0.1]$, we will construct a training process to find best B and R for the experiments. More specially, in a training process, we will try every case of B, R in their corresponding space on the training benchmark and test on other benchmarks, until we find out the values of B and R that can get the best Pearson correlation coefficient on test benchmarks.

Finally, we will give some improvements of the four methods in Equations (4)-(7). On the one hand, since for every concept pair that is not synonym, by Equations (2) and (3), we have $\Im_X(S) = \Im_{RE}(S) = 0$. In this case, we suggest that the weight of Synonym should be ignored since it will weaken the influence of other features. That is, when we choose the mean function as S_{con} in Equation (1), we only compute the mean of $\Im(A), \Im(C), \Im(G)$, i.e., define $S_{con} = (\Im(A) + \Im(C) + \Im(G)) \times \frac{1}{3}$. On the other hand, for methods $SimSec$ and $SimFou$ in Equations (5) (7), if we define $\alpha = 0.5$ in default, then all features have the same influence on the semantic similarity of the compared concepts. Thus, we suggest we should try different values in the test benchmarks to find the most appropriate α in discrete space $[0, 1, 0.1]$. The intuitive idea of trying different values for α is that for methods $SimSec$ and $SimFou$, defining $\alpha = 0$ in function $\Im_{RE}(x, y)$ means we only take the particular features of Con_2 into account and ignore the particular features of Con_1. The bigger the α is, the more particular features of Con_1 are for taking into account. But we do not know whose particular features are more credible. Therefore, we try different values of α.

3.2 Weights assignments with nonlinear fitting

In this subsection, we will consider how to measure the weights of features based on their contribution by the nonlinear fitting.

Firstly, for $\Im_X(x, y)$ in Equation (2), we sort the similarities of four features $\Im(A), \Im(C), \Im(G), \Im(S)$ from small to large, and denote it as $\Im_{X1}, \Im_{X2}, \Im_{X3}, \Im_{X4}$.

Similarly, for $\Im_{RE}(x, y)$ in Equation (3), we can denote the similarities of four features as $\Im_{RE1}, \Im_{RE2}, \Im_{RE3}, \Im_{RE4}$, respectively.

Secondly, we use linear fitting to find best weight $W = (w_1, w_2, w_3, w_4)$ and redefine the four methods in Equations (4)-(7) as follows.

$$SimFirFit(Con_1, Con_2) = w_1 * \Im_{X1} + w_2 * \Im_{X2} + w_3 * \Im_{X3} + w_4 * \Im_{X4}, \quad (10)$$
$$SimSecFit(Con_1, Con_2) = w_1 * \Im_{RE1} + w_2 * \Im_{RE2} + w_3 * \Im_{RE3} + w_4 * \Im_{RE4}. \quad (11)$$

Clearly, we can easily show that the methods in Equations (4)-(7) is a special case of Equations (10) or (11). For instance, if $W = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$, we have $SimFir$ by Equation (10) and $SimSec$ by Equation (11). If $W = (0, 0, 0, 1)$, we have $SimThi$ by Equation (10). Furthermore, by Equations (10) and (11), we can give a formal definition of our method with nonlinear fitting as follows:

Definition 2 (nonlinear fitting method). *Let Con_1 and Con_2 be two concepts defined by four features: Anchor, Category, Gloss, Synonym; $\Im_{X1} < \Im_{X2} < \Im_{X3} < \Im_{X4}$ be the reordered similarities of four features obtained by Equation (2); $\Im_{RE1} < \Im_{RE2} < \Im_{RE3} < \Im_{RE4}$ be the reordered similarities of four features obtained by Equation (3); $W = (w_1, w_2, w_3, w_4)$ be the weights set of the features; and $P = (p_1, p_2, p_3, p_4)$ be the exponent of the similarities of four features. Then the similarity between two concepts Con_1 and Con_2 obtained by weighted feature-based methods with nonlinear fitting can be defined as follows:*

$$SimFirNon(Con_1, Con_2) = w_1 * \Im_{X1}^{p_1} + w_2 * \Im_{X2}^{p_2} + w_3 * \Im_{X3}^{p_3} + w_4 * \Im_{X4}^{p_4}, \quad (12)$$
$$SimSecNon(Con_1, Con_2) = w_1 * \Im_{RE1}^{p_1} + w_2 * \Im_{RE2}^{p_2} + w_3 * \Im_{RE3}^{p_3} + w_4 * \Im_{RE4}^{p_4}. \quad (13)$$

Thus, by Definition 2, in order to obtain the similarity between two concepts by our new methods, we need to select one benchmark as training data to find the power values set P and the weights set W for methods $SimFitNon$ and $SimSecNon$ in discrete spaces, then test on other benchmarks.

3.3 Training process of the parameters

In this subsection, we will show how to use a training process to obtain the power values set P, and the weights set W.

First, when methods $SimFirNon$ and $SimSecNon$ are trained on each benchmark, we denote them as $TestFirTrain$ and $TestSecTrain$. To show the details about training and testing benchmarks, we denote method as $test - benchmark - Fir - train - benchmark$ and $test - benchmark - Sec - train - benchmark$. For instance, method $353tcFirMC30$ is a sub-method of $TestFirTrain$ which means method $SimFitNon$ train on MC30 and test on 353tc. The Pearson correlation coefficient of $TestFirTrain$ and $TestSecTrain$ are defined as the highest Pearson correlation coefficient of all its sub-methods.

In our experiments, we train $SimFirNon$ and $SimSecNon$ in discrete spaces $W \in [-1, 1, 0.02]^4$ and $P \in \{\frac{1}{10}, \frac{1}{9}, \frac{1}{8}, \frac{1}{7}, \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1\}^4$. And Table 1 shows that

the highest Pearson correlation coefficient of method $TestFirTrain$ are 0.726, 0.794, 0.872,0.871,0.525 and method $TestSecTrain$ gets 0.738, 0.801, 0.901, 0.880, 0.548, on benchmark 353tc, Jiang30, MC30, RG65, Sim666 respectively. From Table 1 we can get the following statements:

1. We have not listed the methods that train on Sim666, because it does not achieve any good performance on each benchmark. This is because that Sim666 has many antonyms. For instance, the similarity of "South" and "North" is 0.22 (normalized), but our method $SimFirNew$ get 0.731 for them. When $\alpha = 0$, method $SimSecNew$ get 1.0 for them. Antonyms make so many troubles for researchers. This is why all the methods in Table 2 get the worst Pearson correlation coefficient on sim666. It seems to be that Sim666 is not a good data set for training data.
2. Because the concepts in MC30 is a subset of RG65, so we do not train on one of them and test on the other. Similar to 353tc and Sim666.
3. Although benchmark MC30 and Jiang30 both have only 30 concept pairs, training on MC30 can hardly achieve any good performance while training on Jiang30 can get an improvement in each benchmark. This is because Jiang30 is made for computing semantic similarity based on Wikipedia, especially but MC30 is not. Also, there are many concepts in MC30, and other benchmarks are ambiguous in Wikipedia, but all the concept in Jiang30 come from Wikipedia directly.
4. Table 1 shows that our methods can have better performances on each benchmark by the nonlinear fitting. It turns out to be that only Sim666 is not a good data set for training data, and Jiang30 is the best training benchmark. When we train on it, each method can achieve improvement.

4 Evaluations

In this section, we will summarize all of our methods and compare with previous researches on benchmarks 353tc, Jiang30, MC30, RG65 and Sim666.

In order to make it convenient to describe, we denote a tuple (a, b, c, d, e) to describe the best results of all the methods in one paper. For instance, the best results in [14] are (0.508, 0.443, 0.582, 0.824, 0.381) which means among all the methods in [14], the best Pearson correlation coefficient they can get are 0.508, 0.443, 0.582, 0.824, 0.381 on benchmarks 353tc, Jiang30, MC30, RG65 and Sim666, respectively.

Remarks: the Pearson correlation coefficient of the methods in previous researches in Table 2 are not the same as the Pearson correlation coefficient in their original papers. This is because we apply their methods to the same version of Wikipedia data used in our experiments, instead of their original data in their papers.

In this paper, the best results for each benchmarks we can get are (0.738, 0.801, 0.901, 0.880, 0.548) as shown in Table 2. For methods $TestFirTrain$ and $TestSecTrain$, we choose their best Pearson correlation coefficient in Table 1.

Table 1. Pearson correlation coefficient of methods $SimFirNon, SimSecNon$ train on denser spaces: $W \in [-1, 1, 0.02]^4$ and $P \in \{\frac{1}{10}, \frac{1}{9}, \frac{1}{8}, \frac{1}{7}, \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1\}^4$.

Method	α	POWER	WEIGHT	Pearson
Jiang30Sec353tc	0.2	$(0.125, \frac{1}{9}, 0.5, 1)$	(0.12, -0.1, 0.68, 0.48)	0.777
MC30Fir353tc	-	$(0.1, 0.1, 0.5, 0.5)$	(-0.06, -0.02, 0.34, 0.16)	0.857
MC30Sec353tc	0.2	$(0.125, \frac{1}{9}, 0.5, 1)$	(0.12, -0.1, 0.68, 0.48)	0.864
RG65Fir353tc	-	$(0.1, 0.1, 0.5, 0.5)$	(-0.06, -0.02, 0.34, 0.16)	0.871
RG65Sec353tc	0.3	$(0.125, \frac{1}{9}, 0.5, 1)$	(0.12, -0.1, 0.7, 0.5)	0.880
Sim666Fir353tc	-	$(0.1, 0.1, 0.5, 0.5)$	(-0.06, -0.02, 0.34, 0.16)	0.495
Sim666Sec353tc	0.0	$(0.1, \frac{1}{9}, 0.5, \frac{1}{3})$	(0.0, 0.04, 0.86, 0.36)	0.529
353tcFirJiang30	-	$(\frac{1}{9}, 0.1, 0.5, 1)$	(0.14, -0.16, 0.26, 0.76)	0.685
353tcSecJiang30	0.3	$(\frac{1}{9}, 0.1, 0.5, 1)$	(0.06, -0.24, 0.58, 0.76)	0.714
MC30FirJiang30	-	$(0.1, 0.1, 1, 1)$	(0.06, -0.04, 0.0, 0.24)	0.872
MC30SecJiang30	0.1	$(0.1, 0.1, 1, 1)$	(0.06, -0.04, -0.02, 0.16)	0.901
RG65FirJiang30	-	$(\frac{1}{9}, 0.1, 0.5, 1)$	(0.14, -0.16, 0.26, 0.76)	0.867
RG65SecJiang30	0.5	$(\frac{1}{9}, 0.1, 0.5, 1)$	(0.14, -0.16, 0.26, 0.56)	0.880
Sim666FirJiang30	-	$(0.1, 0.1, \frac{1}{7}, \frac{1}{7})$	(0.1, -0.04, 0.12, 0.3)	0.521
Sim666SecJiang30	0.0	$(0.1, 0.1, \frac{1}{6}, \frac{1}{6})$	(0.22, -0.06, 0.12, 0.6)	0.535
353tcFirMC30	-	$(0.1, \frac{1}{9}, \frac{1}{3}, 0.25)$	(-0.04, -0.08, 0.64, 0.92)	0.698
353tcSecMC30	0.7	$(0.1, 0.1, 1, 1)$	(-0.1, -0.04, 0.94, 0.08)	0.700
Jiang30FirMC30	-	$(0.1, 0.125, 0.1, 1)$	(0.04, -0.02, -0.04, 0.16)	0.794
Jiang30SecMC30	0.0	$(\frac{1}{7}, 0.2, 0.1, 1)$	(0.18, -0.1, -0.16, 0.46)	0.797
Sim666FirMC30	-	$(0.1, \frac{1}{9}, \frac{1}{3}, 0.25)$	(0.0, -0.08, 0.52, 0.98)	0.509
Sim666SecMC30	0.6	$(0.1, 0.125, 1, \frac{1}{3})$	(0.18, -0.42, 0.94, 0.76)	0.469
353tcFirRG65	-	$(1, 1, 0.5, 0.5)$	(-0.1, 0.02, 0.34, 0.14)	0.726
353tcSecRG65	0.2	$(0.1, 0.1, 0.5, 0.5)$	(-0.14, 0.14, 0.98, 0.32)	0.738
Jiang30SecRG65	0.0	$(\frac{1}{7}, 0.125, 0.5, 1)$	(0.06, -0.04, 0.06, 0.1)	0.801
Sim666FirRG65	-	$(0.1, 0.1, 0.25, 0.25)$	(0.08, 0.12, 0.12, 0.44)	0.525
Sim666SecRG65	0.2	$(0.1, 0.1, 0.5, 0.5)$	(-0.14, 0.14, 0.98, 0.32)	0.548

Firstly, we compare with eight traditional semantic similarity feature-based methods in [6], i.e., $SimFir \cdots SimEig$. Their best results are (0.622, 0.781, 0.780, 0.826, 0.469). Obviously, in every benchmark, we get a better result, because our methods overcome their limitations. We remove low-frequency features which contain too much noise. We propose a novel way to compute the similarity of a synonym pair. Both negative and positive weights are used to mitigate incompleteness and validity of Wikipedia. Secondly, we compare with two methods $wpath_{IC_{path}}$ and $wpath_{IC_{corpus}}$ in [14] which measure the semantic similarity between concepts in Knowledge Graphs (KGs) such as WordNet and DBpedia. Semantic similarity is measured by combining the shortest path length between concepts and IC based weight of the shortest path. Their best results are (0.508, 0.443, 0.582, 0.824, 0.381). We also get better results in each benchmark. Thirdly, we compare with three methods $NASARI_{embded}$, $NASARI_{lexical}$ and $NASARI_{unified}$ in [2]. It provides a novel multilingual vector representation of words using structural knowledge from semantic networks with the statistical information derived from text corpora for the effective representation of millions

Table 2. Pearson correlation coefficient of all methods.

English Wikipedia

Method	353tc	Jiang30	MC30	RG65	Sim666
TestFirTrain	0.726	0.794	0.872	0.871	0.525
TestSecTrain	0.738	**0.801**	**0.901**	0.880	0.548
SimFir	0.478	0.749	0.716	0.770	0.434
SimSec	0.513	0.757	0.725	0.780	0.453
SimThi	0.567	0.781	0.756	0.808	0.455
SimFou	0.622	0.763	0.770	0.826	0.469
SimFif	0.302	0.305	0.607	0.507	0.172
SimSix	0.305	0.304	0.610	0.510	0.172
SimSev	0.580	0.727	0.780	0.743	0.423
SimEig	0.550	0.713	0.763	0.724	0.416
$wpath_{IC_{corpus}}$	0.483	0.443	0.515	0.784	0.357
$wpath_{IC_{graph}}$	0.508	0.413	0.582	0.824	0.381
$NASARI_{embeded}$	**0.825**	0.722	0.855	0.849	0.512
$NASARI_{lexical}$	0.807	0.724	0.863	0.861	0.557
$NASARI_{unified}$	0.822	0.722	0.842	**0.881**	**0.574**
$Word2Vec$	0.751	-	0.837	0.820	0.511

German Wikipedia

Method	353tc	RG65	Method	353tc	RG65
TestFirTrain	0.517	**0.731**	$SimFou$	0.514	0.590
TestSecTrain	0.529	0.721	$SimFir$	0.369	0.431
$NASARI_{lexical}$	0.658	0.724	$SimSec$	0.415	0.477
$NASARI_{unified}$	**0.673**	0.685	$SimThi$	0.461	0.523
$Word2Vec$	0.657	0.682	$SimFif$	0.578	0.705
$SimSix$	0.569	0.702	$SimSev$	0.544	0.578
$SimEig$	0.526	0.564			

of BabelNet synsets, including nominal WordNet synsets and all Wikipedia articles. Their best results are (0.825, 0.724, 0.863, 0.881, 0.574). We get better results in Jinag30, MC30 and in RG65 they get 0.881, but we get 0.880. Finally, we compare with Word2Vec [8] where words are represented as vectors. A feed-forward neural network is used to learn continuous representations of words. Word2Vec has two architectures Continuous Bag-of-Words (CBOW) and Continuous Skip-gram Model. Their best results are (0.751, -, 0.837, 0.820, 0.511). They get better results in 353tc, but we get better results in MC30, RG65 and Sim666. We do not run their methods in Jiang30, because there are many concepts in Jiang30 that cannot be represented by vectors. For instance, "Nobel_Prize_in_Literature", "Nobel_Peace_Prize", "Summer_Olympic_Games" and "World_championship" are concepts obtained by combined words and representing words by vectors need to split them.

Besides English Wikipedia, we also try German Wikipedia. While German Wikipedia is so small such that many concepts in Jiang30 and Sim666 do not exist the corresponding concept in it. MC30 is a subset of RG65. Therefore, we only try our methods in German Wikipedia in 353tc and RG65.

The lower part of Table 2 shows the Pearson correlation coefficient of our methods (right part) and the methods in precious researches (left part). When train data, discrete spaces are $W \in [-1, 1, 0.02]^4$, $P \in \{\frac{1}{10}, \frac{1}{9}, \frac{1}{8}, \frac{1}{7}, \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1\}^4$. The best results we can get are (0.529, 0.731). When train data in German Wikipedia, the discrete spaces are the as English Wikipedia. Compared with [6, 2, 8], they all have better performance on 353tc. On the contrary, we have better performance on RG65. This is because 353tc has nearly 200 concept pairs, but RG65 only has about 60. It is too difficult to get a good result if we train on such a small benchmark and test on a much bigger benchmark.

5 Conclusion

It is hard to deal with the incompleteness, invalidity and inconsistency of the knowledge in Wikipedia because anyone can edit a Wikipedia article at any time. Furthermore, in traditional knowledge and feature-based methods for semantic similarity that without weights, all features are treated equally. Each feature makes the same contribution to similarity. But in the weighted methods, it is a complex task to assign weights to features. To address the above issues, in this paper, for the sake of invalidity, we remove low-frequency features (anchors and terms in glosses) in our experiments. Because of incompleteness, we do not limit the sum of all weights to 1. The inconsistency of knowledge has a great influence on the similarity. So when we apply S_{RE} to our methods, we try different values of α, instead of fixing it as 0.5.

Furthermore, we provide a new way to compute the similarity of synonyms. Many previous methods of semantic similarity take the similarity of all synonyms that have common senses as 1 in the score [0,1] when they are applied to Wikipedia. But in our novel methods, we give them an initial similarity and adjust it by the number of their categories. Finally, we provide new methods to compute similarity based on nonlinear fitting. We propose a new way to assign weights to features by training data on one benchmark in discrete space to find weights and powers, then test on others. Compared with previous work, our new methods can always get a better Pearson correlation coefficient on some benchmarks. According to our experiments, Jiang30 is the best benchmark for training data. When we train on it, every method has a better performance on other benchmarks. We also try German Wikipedia. But limited to the size of German Wikipedia, many concepts in benchmarks do not have an article in it. We only try RG65 and 353tc. Training on 353tc has a better performance than previous researches.

In the future, we will try to deal with antonyms. Sim666 is the biggest benchmark among MC30, RG65, Jiang30 and 353tc and training on a bigger data set can always get a better result. But in fact, training on Sim666 is hard to get good results because of antonyms. It is hard to distinguish if a concept pair are antonyms or not by Wikipedia, and they always have many common features. There still lack an efficient way to compute the similarity of two antonyms.

Acknowledgments: The works described in this paper are supported by The National Natural Science Foundation of China under Grant Nos. 61772210 and 61272066; Guangdong Province Universities Pearl River Scholar Funded Scheme (2018); The Project of Science and Technology in Guangzhou in China under Grant No. 201807010043; The key project in universities in Guangdong Province of China under Grant No. 2016KZDXM024.

References

1. Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies*, pages 19–27. Association for Computational Linguistics, 2009.
2. José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64, 2016.
3. Ali Choumane. A semantic similarity-based social information retrieval model. *Social Network Analysis & Mining*, 4(1):1–6, 2014.
4. Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, 2015.
5. Yuncheng Jiang, Wen Bai, Xiaopei Zhang, and Jiaojiao Hu. Wikipedia-based information content and semantic similarity computation. *Information Processing & Management*, 53(1):248–265, 2017.
6. Yuncheng Jiang, Xiaopei Zhang, Yong Tang, and Ruihua Nie. Feature-based approaches to semantic similarity assessment of concepts using wikipedia. *Information Processing & Management*, 51(3):215–234, 2015.
7. Rouzbeh Meymandpour and Joseph G. Davis. A semantic similarity measure for linked data: An information content-based approach. *Knowledge-Based Systems*, 109:276–293, 2016.
8. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Computer Science*, pages 1–12, 2013.
9. George A. Miller and Walter G. Charles. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28, 1991.
10. Euripides G.M. Petrakis, Giannis Varelas, Angelos Hliaoutakis, and Paraskevi Raftopoulou. X-similarity: Computing semantic similarity between concepts from different ontologies. *Journal of Digital Information Management*, 4(4):233–237, 2006.
11. Rong Qu, Yongyi Fang, Wen Bai, and Yuncheng Jiang. Computing semantic similarity based on novel models of semanticrepresentation using wikipedia. *Information Processing & Management*, 54:1002–1021, 2018.
12. Andrea Rodríguez and Max J. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE transactions on knowledge and data engineering*, 15(2):442–456, 2003.
13. Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Communications of the Association for Computing Machinery*, 8(10):627–633, 1965.
14. Ganggao Zhu and Carlos A. Iglesias. Computing semantic similarity of concepts in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):72–85, 2017.