Semantic Keyword Similarity Model

Md. Saiful Islam, Mohammed Eunus Ali, Yong-Bin Kang,
Timos Sellis, and Farhana Murtaza Chowdhury

27 November 2019

1 Introduction

In a social network, each individual's expertise in various fields can be represented by a collection of terms. There are millions of such terms in a large network and it is unlikely for users to come out with the exact terms while raising a query. In this article, we present our proposed *semantic keyword similarity model* which can capture the semantics of terms and keywords in the given attributed graph. Then, we evaluate our model using keywords from ACM taxonomy. This augmented attributed graph has been designed to enrich the coverage and understanding of keywords by associating them with their semantically relevant terms.

2 Semantic keyword similarity model

Given a term, finding its semantically related keywords is not trivial; basic preprocessing like removing whitespaces and stopwords can be helpful but there are still some major concerns. Two or multiples terms with a slightly syntactical difference can indicate a similar or the same keyword (e.g., "error detection and error correction" and "error detection and correction"). Even two different terms can represent the same keyword (e.g., "AI" and "artificial intelligence"). Also, some different terms are semantically very related to each other (e.g., "neural network" and "gradient descent optimization").

We first train a *word embedding* model to generate an embedding vector of any given term (and also keyword). Then, we use such a vector to capture the semantics of a term and find its relevant keywords. For training this model, we collect a domain corpus (e.g. scientific publications in an academic domain), where each document is already associated with keywords. We call this *training corpus* in this section. Among the keywords in the training corpus, we select N keywords which have the N-top document frequencies (a document frequency is defined as the number of times a keyword appears in all the documents in the corpus). Then, our focus is to develop a model that can find the relevant keywords out of these N-top keywords given a term by using the word embedding model.

Word embedding is a learning technique that can enable words or phrases to map to vectors of real number. Word2vec [5] is a representative word embedding model that can be trained to construct linguistic contexts of words and thereby generate such a mapping. The output of Word2vec is a vector space where similar words are positioned close to one another. It has been well demonstrated that Word2vec has many advantages over earlier traditional embedding techniques for semantic analysis of words[5].

First, to build a Word2vec model on the training corpus, we perform tokenization from each text document on the training corpus. This process includes removing stopwords, and extracting only nouns, adjectives and gerunds as usually important concepts are represented via nouns [4]. Also, some keywords consist of adjectives and gerunds (i.e., **expert** systems, machine **learning**, etc.). After tokenization, we convert each of the extracted words into lemmatized form to identify its single canonical form.

Second, we feed the extracted words into a Word2Vec model that represents each word as a vector. Using similarities between two vectors, we can estimate how close and distant they are in terms of semantic meaning.

Third, we represent a keyword or term as a vector. A keyword/term can be thought of as a phrase containing one or many words. In our approach, the representative vector is formed using the average of the embedding vectors of the constituent words. By doing so, we are now ready to estimate the similarity between any term and any keyword according to their vectors.

Given any two terms (two keywords, or a term and a keyword) t_1 and t_2, we denote their embedding vectors as x_{t_1} and x_{t_2}, respectively. In the following, we present three approaches to estimating a similarity between these two terms, denoted as $S(t_1, t_2)$:

- **cosine**
  $S(t_1, t_2)$ is calculated by the cosine similarity of their embedding vectors x_{t_1}, x_{t_2}:
  $$S(t_1, t_2) = cosine(x_{t_1}, x_{t_2}) \tag{1}$$

- **normalized maximum**
  Given two terms t_1 and t_2, we generate their semantically relevant vectors V^{t_1} and V^{t_2}, respectively. Given a term t, its vector V^t is denoted as $V^t = [(w_1^t, s_1^t), (w_2^t, s_2^t), \cdots, (w_L^t, s_L^t)]$, where w_i^t is the embedding vector of i^{th} most similar word to x_t and s_i^t is the corresponding similarity score, and L is the number of the similar terms of t. Now, $S(t_1, t_2)$ is formulated using their corresponding V^{t_1} and V^{t_2} motivated by the similarity measure [1]:
  $$S(t_1, t_2) = \frac{\sum\limits_{(w_1, s_1) \in V^{t_1}} sim_m(w_1, V^{t_2}) + \sum\limits_{(w_2, s_2) \in V^{t_2}} sim_m(w_2, V^{t_1})}{|V^{t_1}| + |V^{t_2}|} \tag{2}$$
  Here, $sim_m(w, V^t) = \max_{(w_i, s_i) \in V^t} sim(w, w_i)$ and $sim(w, w_i)$ can be calculated by measuring the cosine similarity of the embedding vectors of w and w_i.

- **indirect cosine**
  From V^{t_1}, V^{t_2}, we construct a vocabulary, U combining words using the union operation. Formally,
  $$U = \{w : (w, s) \in V^{t_1} \cup (w, s) \in V^{t_2}\} \tag{3}$$
  To simplify our notation, let U be denoted as $U = \{w_1, w_2, \cdots, w_n\}$, where $n = |U|$. Now, we define another vector $SV^t = [s_1, s_2, \cdots, s_n]$, where s_i is the similarity score of term t to word $w_i \in U$, which can be found from V^t. If $(w_i, s_i) \notin V^t$, s_i is set to 0. Then, $S(t_1, t_2)$ is calculated using their SV^{t_1}, SV^{t_2}:
  $$S(t_1, t_2) = cosine(SV^{t_1}, SV^{t_2}) \tag{4}$$

In our evaluation in Section 3, we compare the above three methods and choose the best one to augment the attributed graph. Finally, for any term t, we calculate its similarity with all the keywords in the given attributed graph, and find M-top most relevant keywords (if exist) ranked based on the similarity scores. M is a system configurable parameter. By default, M is set to 10.

3 Evaluation of semantic similarity model

In this section, we first present the experimental studies to evaluate our proposed similarity metrics followed by parameter selection for the semantic similarity model.

In Section 2, we presented three similarity metrics for finding semantically related keywords or terms given a term or keyword. We now empirically evaluate which metric is the most appropriate. Our goal here is to measure a similarity between two terms using the three measures, and compare the similarity results with the ground truth.

As the ground truth, we use the semantic similarity measure that has been widely used in Information Retrieval for measuring the similarity between two concepts in a given taxonomy [6]. Its unique feature is measuring the similarity based on the intrinsic information content of the taxonomy structure only. As the taxonomy, we use the ACM taxonomy provided by the 2012 ACM Computing Classification System[1] that contains 2,113 topics and organizes them hierarchically based on relevance (e.g., “clustering" is a sub-topic of “data mining").

In our context, each topic in the taxonomy can be seen as a keyword or a term, and each of our proposed similarity measures can be considered as a ranker that finds the most similar topics to any topic in the taxonomy. So, we use a widely accepted performance measure of ranking systems, that is, Normalized Discounted Cumulative Gain ($NDCG$) [7], to evaluate the proposed similarity metrics.

First, given the taxonomy τ, we give the ground truth formula of the similarity between two topics t_1 and t_2 proposed in [6]: $sim_{jcn}(t_1, t_2) = 1 - [d^{\tau}_{jcn}(t_1, t_2)/2]$, where $d^{\tau}_{jcn}(t_1, t_2)$ denotes a distance metric between t_1 and t_2 as $d^{\tau}_{jcn}(t_1, t_2) = IC^{\tau}(t_1) + IC^{\tau}(t_2) - 2 \times IC^{\tau}(lcs(t_1, t_2))$ [3]. Here, $lcs(t_1, t_2)$ is the “least common subsumer" in τ that subsumes t_1 and t_2. Finally, $IC^{\tau}(t)$ indicates the information content of topic t in τ, calculated by $\log\left(\frac{|sc^{\tau}(t)|+1}{|\tau|}\right) / \log\left(\frac{1}{|\tau|}\right)$ as in [6]. Here, $sc^{\tau}(t)$ is the set of subsumed topics of t and $|\tau|$ is the total number of topics in τ.

Second, given a topic t in τ, our task is to rank the M-top similar topics $\{w_1, w_2, \cdots, w_M\}$ in τ. The ranking goodness is evaluated by $NDCG@M$ defined as $NDCG = \frac{1}{|\tau|}\sum_{t \in \tau} NDCG^t$, where $NDCG^t = \frac{DCG^t}{IDCG^t}$. Here $DCG^t = \sum_{i=1}^{M} \frac{sim_{jcn}(t, w_i)}{\log(i+1)}$ is the discounted cumulative gain that gives more importance on the ranking of a more relevant entity than the ranking of entities with lower relevance. $IDCG^t$ provides the maximum DCG value that could be obtained by a ranker if the ranking was ideal.

To obtain a vector representation of each topic in τ, we used Google's pre-trained word2vec model[2]. This model includes embedding vectors for a vocabulary of 3 million words and is trained on about 100 billion words from a Google News dataset which also covers academic research. The length of the embedding vector is 300.

Table 1 shows the comparison among the three similarity metrics in terms of $NDCG@50$, $NDCG@20$, $NDCG@10$. As seen, overall, it turns out that `indirect cosine` outperforms the other metrics. Thereby, we

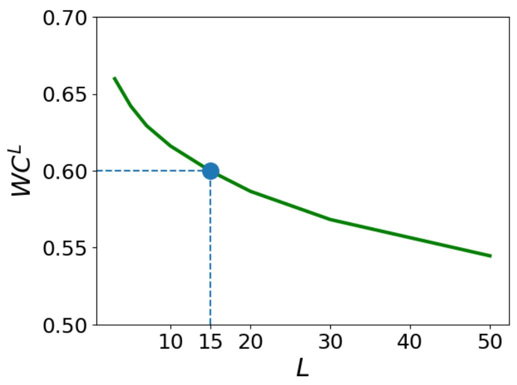
[1]`https://dl.acm.org/ccs/ccs_flat.cfm`
[2]https://code.google.com/archive/p/word2vec/

have chosen this metric to associate each term or keyword with its semantically related keywords or terms.

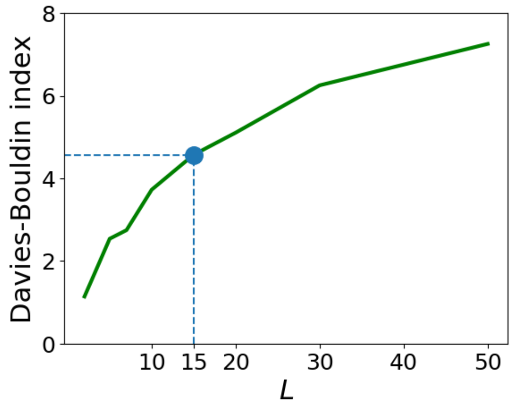
Metric	**cosine**	**normalized maximum**	**indirect cosine**
NDCG@50	0.528	**0.543**	0.542
NDCG@20	0.537	0.583	**0.607**
NDCG@10	0.573	0.614	**0.633**

Table 1: Performance results of the three similarity metrics

In the `indirect cosine` similarity measure, given a term t, we need to retrieve its semantically relevant vector $V^t = [(w_1^t, s_1^t), (w_2^t, s_2^t), \cdots, (w_L^t, s_L^t)]$ (see Section 2). V^t contains L-top words and their similarity to t. Here, a challenge is how to determine a good value for L. To examine this, we use two measures. The first is *word coherence,* $WC^L(V^t)$, to evaluate how coherent the L-top words in vector V^t are. The more coherent it is, the better we can represent the set of keywords. We define the word coherence as the average pairwise cosine similarity of the L-top words. Formally, $WC^L(V^t) = \frac{2}{L \times (L-1)} \sum_{i=1}^{L} \sum_{j=i+1}^{L} sim(w_i, w_j)$, where $sim(w_i, w_j)$ is the cosine similarity of the embedding vectors of w_i and w_j. $WC^L = 0$ indicates no coherence and value $WC^L = 1$ indicates maximum coherence. Also, in a sense, each term is a cluster containing its relevant words. As the second measure, we use *Davies-Bouldin Index* [2] that optimizes two criteria: (1) minimizing intra-distance between words and the centroid, and (2) maximizing inter-distance between keywords. Values closer to zero indicate a better clustering.



(a) Average word coherence WC^L



(b) Davies-Bouldin index

Figure 1: L=15 (L: number of top words) provides reasonably high keyword coherence and low Davies-Bouldin index

L determines the number of similar words. A small value usually contains words with high coherence and better clustering, but it is tough to find similar keywords.A large value contains words with low coherence and worse clustering, but it is easier to find similar keywords. So, we need to choose a value which provides a good trade-off. The desired value of L should be able to find a sufficient number of similar keywords, and provide high word coherence WC^L and low Davies-

Bouldin index. Figure 1 shows the average cohesiveness and Davies-Bouldin index vs. various values of L. Here $L = 15$ provides an excellent trade-off being able to retrieve at least 10 similar keywords given a term in 98% cases.

References

[1] Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. The evaluation of sentence similarity measures. In Il-Yeol Song, Johann Eder, and Tho Manh Nguyen, editors, *Data Warehousing and Knowledge Discovery*, pages 305–316, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[2] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[3] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

[4] Yong-Bin Kang, Pari Delir Haghighi, and Frada Burstein. Cfinder: An intelligent key concept finder from text for ontology development. *Expert Syst. Appl.*, 41(9):4494–4504, July 2014.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[6] Nuno Seco, Tony Veale, and Jer Hayes. An intrinsic information content metric for semantic similarity in wordnet. In *Ecai*, volume 16, page 1089, 2004.

[7] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. A theoretical analysis of ndcg type ranking measures. In *Conference on Learning Theory*, pages 25–54, 2013.