Knowledge-Based Systems 188 (2020) 105051

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

On Computing Entity Relatedness in Wikipedia, with Applications☆

Marco Ponza [a,*], Paolo Ferragina [a], Soumen Chakrabarti [b]

[a] University of Pisa, Italy
[b] IIT Bombay, India

ARTICLE INFO

Article history:
Received 5 April 2019
Received in revised form 16 September 2019
Accepted 18 September 2019
Available online 20 September 2019

Keywords:
Entity relatedness
Wikipedia
Knowledge graph

ABSTRACT

Many text mining tasks, such as clustering, classification, retrieval, and named entity linking, benefit from a measure of relatedness between entities in a knowledge graph. We present a thorough study of all entity relatedness measures in recent literature based on Wikipedia as the knowledge graph. To facilitate this study, we introduce a new dataset with human judgments of entity relatedness. No clear dominance is seen between measures based on textual similarity and graph proximity. Some of the better measures involve expensive global graph computations. We propose a new, space-efficient, computationally lightweight, two-stage framework for relatedness computation. In the first stage, a small weighted subgraph is dynamically grown around the two query entities; in the second stage, relatedness is derived based on computations on this subgraph. Our system shows better agreement with human judgment than existing proposals both on the new dataset and on an established one. Our framework also shows improvements with respect to the state-of-the-art on three different extrinsic evaluations in the domains of ranking entity pairs, entity linking, and synonym extraction.

1. Introduction

The use of knowledge graphs (KGs) has proliferated in text mining and search tasks, ranging from clustering, classification and retrieval in long texts – see, e.g., [1–4] – to short posts, news and queries — see, e.g., [5–7]. Most of these consumers of KGs need a measure of relatedness between entities in the KG. Consequently, there has been a series of proposed relatedness measures, mainly based on WordNet or Wikipedia. Some of these have been used in downstream applications, leading to their *extrinsic* evaluation [8]. In contrast, *intrinsic* evaluation against human judgments of relatedness has been rare and confined mainly to word pairs – see, e.g., [9–12] – thus, we know of no significant dataset of human-generated relatedness scores between entities in a large KG.

Toward deeper understanding of all the well-known relatedness measures, we introduce WiRe, a new dataset of 503 pairs of entities occurring in Wikipedia and drawn from the New York Times dataset [13] with human-assigned relatedness scores. This is in contrast to *word* similarity datasets such as WikiSim [10], commonly used in NLP (although there, too, lexical networks can provide some graphical signals to infer relatedness).

Using both WiRe and WikiSim, we present a thorough, systematic study of essentially all relatedness measures known from recent literature. Some have been designed specifically for the entity relatedness task, whereas others will be adapted in this paper for it. The measures can be roughly divided into ones that use text similarity, and ones that use graph proximity. Far more effort has been spent on graph-based relatedness, which includes the measure introduced by Milne and Witten [10], widely used in the best entity linkers [5,8,14–17]. Somewhat surprisingly, we found no overwhelming winner by measuring quality using Pearson and Spearman correlations against human judgments. In addition, we found some of the best global relatedness measures too slow to execute on large KGs.

In the quest for a practical, time- and space-efficient, robust and more accurate relatedness measure, we propose here a two-stage framework. Using a selection of known measures, we grow a small subgraph around each entity of the query pair, and then use further selection or combination of known relatedness measures to compute the edge weights in that subgraph. By populating this new framework with various choices for the first and second stages, we come up with a system that shows significantly better agreement with human judgment than previous relatedness measures. To the best of our knowledge, we are the first to report on an intrinsic evaluation of textual similarity and graph proximity measures applied to the entity relatedness task, as well as to provide a configurable joint framework without any need for further feature engineering.

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.105051.

* Corresponding author.
E-mail address: marco.ponza@di.unipi.it (M. Ponza).

https://doi.org/10.1016/j.knosys.2019.105051
0950-7051/

We supplement the above intrinsic evaluation with three different extrinsic evaluations in several applications that respectively lie in the domains of entity ranking pairs, entity linking, and synonym extraction. The effectiveness of our framework is shown with the improvement of the results with respect to known state-of-the-art solutions.
To summarize, our contributions are:

1. A new entity relatedness dataset, WiRe, comprising judgments by human experts on 503 pairs of named entities occurring in Wikipedia.
2. A new, two-stage, fast and space-efficient entity relatedness framework that returns more accurate scores than prior proposals.
3. A comprehensive study involving an intrinsic evaluation of our two-stage framework and a number of relatedness measures proposed in recent literature over the new WiRe and the known WikiSim datasets, properly complemented by three different extrinsic evaluations in the domains of (i) entity linking, (ii) ranking pairs of entities, and (iii) word similarity.
4. Publicly available WiRe data and code of all tested algorithms.[1]

2. Terminology

We will model Wikipedia as a labeled graph, in which nodes represent Wikipedia pages describing entities, edges correspond to hyperlinks between Wikipedia pages. We are primarily interested in characterizing similarity between entities. We will regard Wikipedia as mapping from each entity ID to the text content of its definition page. The Wikipedia knowledge graph (KG) structure is the set of all hyperlinks connecting Wikipedia's pages. We will use $I(u)$ and $O(u)$ to denote the in- and out-neighbors of the node u in the KG, respectively, and denote by $\Gamma(u) = I(u) \cup O(u)$ the undirected neighborhood of u.

3. Related work

Recent literature offers many algorithms for estimating relatedness of words in the WordNet graph (see Agirre et al. [11] and references therein). Words are ambiguous lexical elements. WordNet contains few proper names, neologisms, and domain-specific technical words compared to generic concepts. Its graph is much smaller than the Wikipedia graph. WordNet was carefully crafted by a small team of linguists working closely together, whereas Wikipedia is written and maintained 'organically' by a large number of editors with no central coordination. For all these reasons, our problem is substantially different, in terms of both the semantics of relatedness, and the space–time efficiency problems that we need to address.

The focus of this paper is the design of *unsupervised* methods for estimating the relatedness between Wikipedia entities, which are unambiguous semantic concepts. In our study we will test many known algorithms which use either the textual content of the Wikipedia pages or the structure of the Wikipedia graph, and combinations thereof. We mention that, among other algorithms, we will test the best-known and most popular ones such as ESA [9], the method proposed by Milne and Witten [10], and the most recent approaches based on entity embeddings [1].

The literature offers other approaches to entity relatedness which are either *supervised* and applied to learning-to-rank frameworks [18], or deploy external/additional sources of information such as temporal, categorical, crawled (Web) documents, or clickstream information [11,12,19,20]. Comparison with these approaches is out of the scope of this work.

The following of this section is focused on the review of a large number of relatedness methods which have been previously proposed in the literature to estimate the "relatedness" between pairs of nodes in a graph. Most of these methods were devised in contexts which are different from the one we deal with in this paper: document annotation [21], word and document similarity [9], personalized Web search [22], machine translation [23], document classification [24,25], and link prediction [26]. However, they show similarities with the problem at hand and are adaptable to the entity relatedness problem over KGs. For ease of exposition, we cluster known relatedness methods into two categories. The first one includes methods which focus on the textual information in the corpus describing the entities (Section 3.1), whereas the second one includes methods which use the hyperlinked structure of the graph connecting the entities (Section 3.2).

3.1. Relatedness based on corpus text

These methods range from classical occurrence-based algorithms (such as VSM and ESA) to modern techniques based on embeddings (such as Entity2Vec). All share the goal of modeling the textual content of a document (that defines an entity) with a vector of real numbers, which is then used, in place of the (possibly long) document, to estimate the relatedness with other entities (also vectors) via classic geometric measures. We sketch below the methods we will evaluate in Section 6.

3.1.1. Sparse word counts

These methods rely on statistical models built upon the occurrences or co-occurrences of terms in the corpus. We can identify three main approaches.

Vector Space Model (VSM). In this classical representation, a document d is modeled as a sparse vector over terms t weighted according the well-known TF–IDF score of t in d. These vectors are normalized to unit length, and relatedness between two entities is computed as the dot-product (equivalently, cosine of the angle) between their corresponding vectors.

Explicit Semantic Analysis (ESA) [9]. While VSM builds document representations, ESA builds term representations based on documents where they occur, again using TF–IDF weights. To adapt this to entity relatedness, we use Wikipedia documents with gold mentions of each entity. Cosine is again frequently used to measure similarity between a pair of ESA vectors.

Language Models (LM) [27]. In LM, we derive the probability of possible word sequences by aggregating counts from the corpus and estimating the relatedness between two entities with the Kneser–Ney function. See Pauls and Klein [27] for details.

3.1.2. Methods based on word embeddings

In this recently invented family of methods, words and documents are modeled via their latent embeddings which are learned from the input corpus using neural networks. In this paper we will consider only the main approaches, adapting some of them to work in our "entity relatedness context" as explained below.

Latent Dirichlet Allocation (LDA) [28]. LDA is a popular generative probabilistic model that fits k latent topics, each defined via a multinomial distribution over the corpus's vocabulary. After a learning phase, where the topics distributions are learned from

[1] github.com/mponza/WikipediaRelatedness

Table 1
Summary of relatedness measures based on neighbor nodes.

Method	Equation
Adamic–Adar	$\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log(\lvert\Gamma(w)\rvert)}$
Bibliographic Coupling	$\lvert O(u) \cap O(v)\rvert$
Co-Citation	$\lvert I(u) \cap I(v)\rvert$
Common Neighbors	$\lvert\Gamma(u) \cap \Gamma(v)\rvert$
Dice	$2\lvert\Gamma(u) \cap \Gamma(v)\rvert \,/\, (\lvert\Gamma(u)\rvert + \lvert\Gamma(v)\rvert)$
Jaccard	$\lvert\Gamma(u) \cap \Gamma(v)\rvert \,/\, \lvert\Gamma(u) \cup \Gamma(v)\rvert$
Milne&Witten	$1 - \frac{\log(\max(\lvert I(u)\rvert, \lvert I(v)\rvert)) - \log(\lvert I(u) \cap I(v)\rvert))}{\log(\lvert V\rvert) - \log(\min\{\lvert I(u)\rvert, \lvert I(v)\rvert\})}$
Overlap	$\lvert\Gamma(u) \cap \Gamma(v)\rvert \,/\, \min\{\lvert\Gamma(u)\rvert, \lvert\Gamma(v)\rvert\}$

the input corpus, LDA can map documents to a vector of weights (the "embeddings") over these latent topics. Entities are compared by computing the cosine between the topic distribution weights of their definition documents.

Entity2Vec [1]. This is the application of Word2Vec [29] to the entities of Wikipedia. The idea is to turn every Wikipedia page into a sequence of entity IDs by substituting every hyperlink with the ID of the destination page, and then computing the embeddings of those entity IDs by processing that sequence via CBOW or Skip-gram models. The relatedness between two entities is eventually computed as the cosine between their embedding vectors.

E5 System [30]. This system jointly learns entity and words representations in the same latent space via neural network through the use of a Skip-gram model applied over two different textual corpora: (1) the Wikipedia corpus where hyperlinks are replaced by proper placeholders (as done by Entity2Vec), (2) the previous Wikipedia corpus in which words are removed and only placeholders are left. The final relatedness score is computed as the arithmetic mean between the average of multiple cosine-similarities computed by different embedding models and the Milne&Witten relatedness measure [10]. Because this system has been proposed after our original work [31] and experimented over the same datasets, we will compare our framework against it in a dedicated paragraph.

3.2. Relatedness based on graph structure

More directly related to our goal are prior methods to characterize node-to-node similarity in (possibly weighted and directed) graphs, based on the neighborhood of those nodes (or potentially the whole graph). In this section, we survey well-known graph-based relatedness measures, and how to adapt them to our task. Table 1 reports the ones based on immediate neighbors.

3.2.1. Relatedness based on random walks

All proposals in this family argue that two nodes u, v are related if two random walkers, started respectively on u and v, frequently encounter the same nodes. We have experimented with the best-performing methods in this family according to the results published in the literature, and adapted them to work on the Wikipedia graph.

PPR + Cos [22]. In topic-sensitive or personalized PageRank, a random surfer intermittently *teleports* back to a subset of nodes while walking on graph edges. In the extreme case, teleport is deterministically set to one node x, the resulting PageRank vector $p(u)$, which is a form of probabilistic signature of reachability from node u to all other nodes. As a result, the relatedness between two queried nodes u and v can then be computed by the cosine similarity between $p(u)$ and $p(v)$. This approach needs as many PageRank computations as queried nodes u (for faster approximations see, e.g., Maehara et al. [32]).

CoSimRank [23]. This is a recent enhancement to PPR + Cos introduced to find synonyms and translations of words. It combines features of SimRank [33] and PPR + Cos. The key idea is that early meetings at nodes during the two random walks from u and v are more valuable than later meetings. In our experiments we omit SimRank and variants [33,34] because CoSimRank was established to be better in terms of time/space complexity and accuracy [23].

Katz Relatedness [35,36]. It is a relatedness measure expressed as the average number of shortest paths at different lengths, properly scaled by a weight decay [36]. Unfortunately, this technique presents scalability issues [36] making infeasible to apply it over large-scale knowledge graphs. On the other hand, a recent work [36] has shown that a random walk re-formulation (i.e., PageRank with no use of the teleport vector) of the Katz relatedness achieves very close performance to the original one [36]. Accordingly, we will focus our experiments on *pure* random walks methods (i.e., PPR + Cos and CoSimRank) thus implicitly testing the results based on this approach.

WikiWalk [37]. This hybrid approach applies PPR + Cos while using the ESA vector of nodes to bias the teleportation jump of the random walk.

Commute Time [38]. The Commute Time between nodes u, v is defined as the average number of steps that a random walk, starting at node u, takes to reach node v for the first time and then come back to u. It is a distance metric on a graph, and can be computed via the pseudo-inverse of the Laplacian, L^+, of the graph.

3.2.2. Relatedness based on graph embeddings

The following methods have not been used for estimating entity relatedness in KGs. We include them because they are effective in estimating node similarity in other graphs for other applications. We will concentrate on DeepWalk which learns high quality representations [25]. Therefore, in our experiments, we will omit SVD, LINE and Commute Time.

DeepWalk [25]. It performs a random walk from a focus node, visiting other context nodes. Focus and context nodes fulfill the same purpose as focus and context words in word embeddings. At this point, CBOW or Skip-gram can be run to find embeddings for all nodes. The intuition is that similar nodes should generate similar paths and thus embed to similar vectors.

LINE [24]. It is a variant of DeepWalk specifically designed to work on directed and weighted graphs and to preserve both first- and second-order proximities between nodes. Our experiments have shown that its performance on our datasets is poor and thus this method will be omitted from tables to save space.

Singular Value Decomposition (SVD) [39]. Levy and Goldberg [40] showed that Word2Vec is equivalent to factorizing a matrix that is a function of word-word co-occurrence counts. The triangle between text, Word2Vec and matrix factorization is in fact closed by classic application of spectral analysis on the adjacency matrix of a graph, embedding each node to a singular vector. Relatedness is again measured as cosine between the singular vectors.

Table 2
WiRe dataset statistics bucketed according to human assessed relatedness score.

Pairs Type	Human relatedness		
	[0, 3]	(3, 6]	(6, 10]
(Salient, Salient)	25	57	90
(Nonsalient, Salient)	44	48	79
(Nonsalient, Nonsalient)	41	31	88

Table 3
WiRe dataset statistics bucketed according to the type of entity.

Pairs type	Entity type			
	Person	Location	Organization	Other
(Salient, Salient)	79	90	89	86
(Nonsalient, Salient)	78	87	89	88
(Nonsalient, Nonsalient)	73	67	71	109

Table 4
WiRe dataset statistics bucketed according to their distance in the Wikipedia Graph.

Pairs type	Distance		
	1	2	3
(Salient, Salient)	90	81	1
(Nonsalient, Salient)	82	61	28
(Nonsalient, Nonsalient)	75	58	27

4. The WiRe dataset

In this section we describe the process we have adopted to create our novel **Wi**kipedia-based entity-**Re**latedness dataset, WiRe. It complements the well-known WikiSim dataset [10] with pairs of *named entities* and associated relatedness scores assigned by a pool of human assessors. WikiSim was built by manually adapting the original WordSim-353 dataset (commonly used for evaluating word-similarity algorithms) to the entity relatedness task. In contrast, WiRe has been devised for benchmarking entity relatedness solutions with a larger set of entities, properly selected and evaluated via a three-phase procedure. The goal is to carefully balance several *coverage* issues which are crucial for our comparative analysis: *salience* and *co-occurrence* of entities, their *type* and *distance*[2] in the KG (Phases 1 and 2 below).

The outcome are 503 pairs of carefully-chosen named entities with intrinsic relatedness judgments from three experts who based their evaluation on the textual description of the Wikipedia entities and further investigation of their relationships by possibly taking advantage of other sources (details on Phase 3 below).

4.1. Phase 1: Generation of many entity pairs

We drew entity pairs from the dataset published by Dunietz and Gillick [13], which has 2 203 909 entity annotations within about 100 000 news articles drawn from the New York Times Corpus (NYTC). An annotation is a tuple (*entity*-ID, *news*-ID, *salience*), where *entity*-ID (resp., *news*-ID) is the entity (resp., news) identifier and *salience* is a binary label which reflects the centrality of *entity*-ID in *news*-ID, computed with a rule-based algorithm [13].

For each news article, we generated all entity pairs, and then grouped them in three classes: (Salient, Salient) pairs, (Nonsalient, Salient) pairs, and (Nonsalient, Nonsalient) pairs. Then we refined each class by discarding those pairs whose co-occurrence frequency in NYTC is ≤ 10. Fig. 1 shows a graphical example of this process. After this step we got 2892, 36 528 and 145 163 pairs of entities, respectively for the (Salient, Salient), (Nonsalient, Salient) and (Nonsalient, Nonsalient) sets. Fig. 1 shows.

4.2. Phase 2: Selection of pairs satisfying coverage requirements

Next we select a subset of interesting entity pairs that match the *coverage requirements* sketched above. More precisely, each one of the three entity-pair classes [i.e., (Salient, Salient), (Nonsalient, Salient) and (Nonsalient, Nonsalient)] is subdivided using the following properties:

- co-occurrence frequency of the entity pair, by considering the three ranges: (10, 15), [15, 25) and ≥ 25;
- type of the entities: Person, Location, Organization and Other (i.e., none of the previous);
- the distance between two entities, which is at most 3 in the dataset.

The above bucketing is to guarantee sufficient diversity and fairness in our evaluation. Finally, to achieve balance among these buckets and avoid over-representation of some frequent entities, we sampled three pairs of entities from each subclass guaranteeing that each entity appears less than 3 times, overall. After this stage, WiRe consists of a total of 503 entity pairs: 172 of type (Salient, Salient), 171 of type (Nonsalient, Salient) and 160 of type (Nonsalient, Nonsalient).

4.3. Phase 3: Generating ground-truth scores

Two human assessors were required to assign a relatedness (integer) score in the range [0, 10] to each entity pair identified in Phase 2. The annotation procedure was conducted *independently* by each assessor: (1) reading the Wikipedia article of each entity in the evaluated pair and, (2) possibly searching Wikipedia, if one entity is not mentioned by the other entity's page, looking for the existence of any relation between them. Each human assessor thus assigned a relatedness score to the pair. If their scores coincided then this was taken as the ground-truth for that pair, otherwise a third evaluator arrived at a reconciled score through discussions.

The Kendall's τ between the final ground-truth and the independent annotations provided by the two experts was very high: 0.80 and 0.77, respectively. Tables 2–4 report some statistics over this dataset. It is interesting to note that the relatedness score is not sensitive to how salient the entities were in the news articles mentioning them. Overall, we believe (and corroborate with the entity linking application, in Section 6.8) that WiRe's intrinsic evaluation will correlate well with extrinsic applications.

5. Our two-stage framework

Our framework for computing entity relatedness combines textual context and the KG structure in an efficient and efficacious way. We will first describe it, and then we will instantiate various components of the framework with concrete algorithmic choices, thus offering a wide spectrum of approaches to solve the relatedness problem faster and better. The experiments of Section 6 will prove that our approach is much faster and improves the accuracy of the known methods by a large margin.

Our framework works in two stages: the first stage consists of three steps which create a small weighted subgraph grown around the two query entities. The second stage derives the relatedness between the query entities based on computations on this small subgraph. Fig. 2 shows a graphical representation of our two-stage framework, and whose algorithmic details are described in the following sections.

[2] In this paper, unless otherwise specified, the distance between nodes is measured as the unweighted length of the shortest path in the undirected KG.

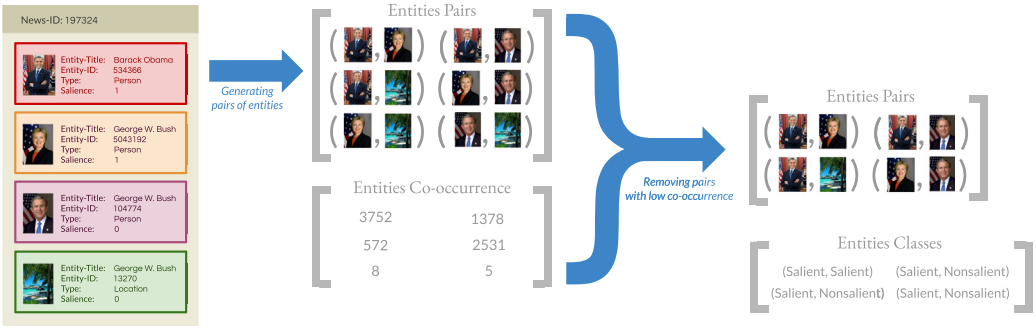


Fig. 1. Generation of pairs of entities given an input document which contains a set of entities annotated by Dunietz and Gillick [13]. First, we generate a list of entity pairs (the order of the entities in the pair is not relevant) and, second, we refine this list by discarding the pairs with a low co-occurrence frequency. Entity co-occurrence is computed over the whole NYTC dataset.

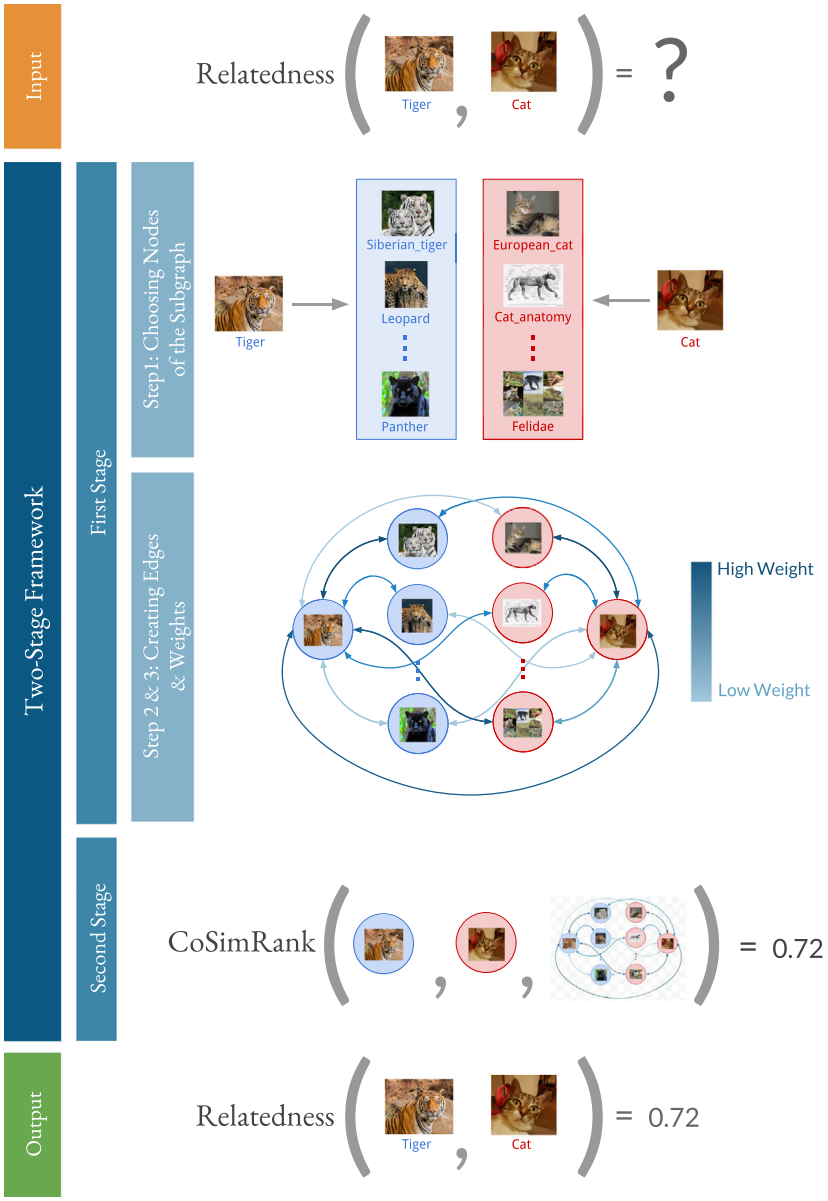


Fig. 2. Example of the two-stage framework on two query entities Tiger and Cat.

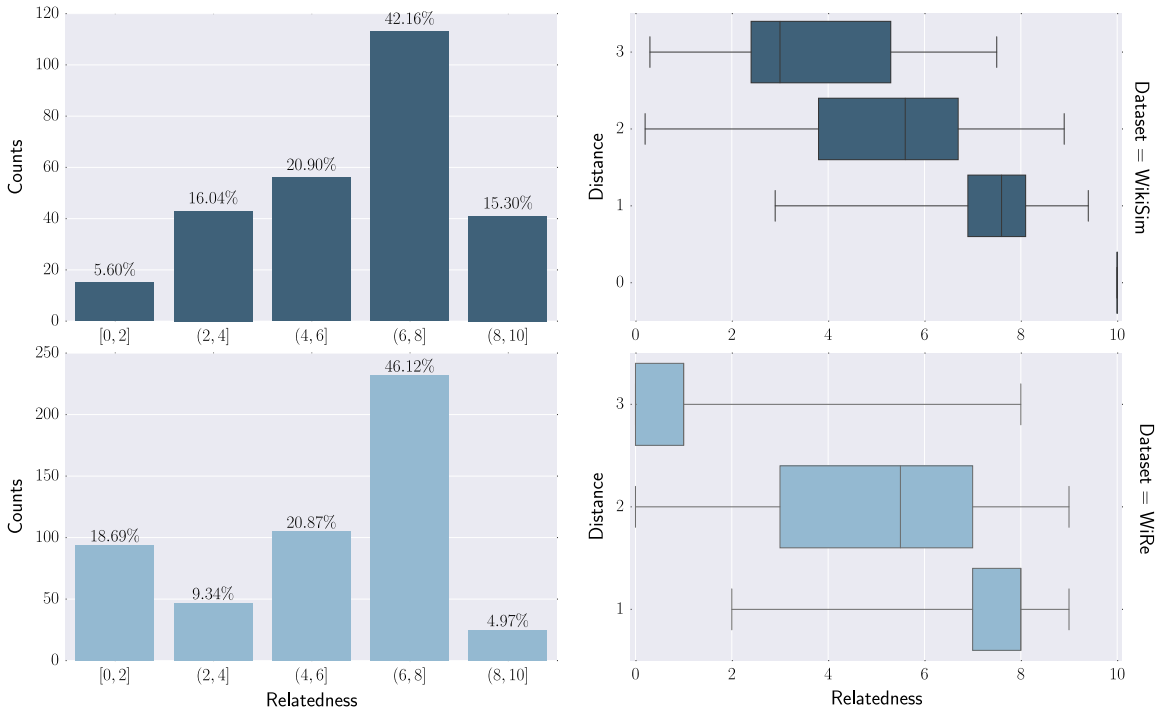
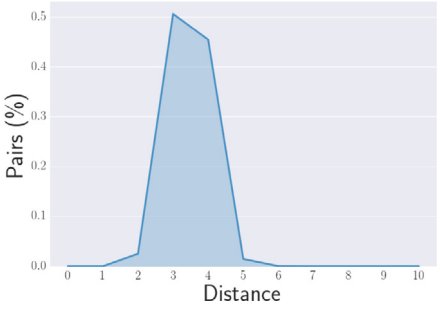


Fig. 3. Data characteristics (top: WikiSim, bottom: WiRe). Charts in the left column show counts of instance pairs with relatedness in bucketed ranges. Most pairs are moderately related. Charts in the right column show distance distribution against bucketed relatedness. Generally relatedness reduces with graph distance, but there is a broad spread.



Statistics	Value
—Nodes—	4 730 474
—Edges—	97 718 760
—CCs—	64
Largest CC	4 730 329
Diameter	15
AVG Distance	3.46
Spid	0.09

Fig. 4. Probability mass function of the distance distribution (left) and several statistics (right) computed on the Wikipedia graph.

5.1. First stage: Subgraph selection

Step 1: Choosing Nodes of Subgraph G_C. Given the queried entities u and v, we use one of the methods described in the previous sections to derive the top-k entities in the KG related to u (call them R_u), and the top-k entities in the KG related to v (call them R_v). In our testbed, we apply the best methods that deploy the textual content of Wikipedia (i.e., ESA and Entity2Vec) and the best methods that deploy its hyperlinked structure (i.e., DeepWalk or Milne&Witten). Algorithm 1 shows the pseudocode implementing the construction of the node set V of G_C. The value of k will be tested in 10–50 in the experimental section; this is of negligible size compared to a typical KG, thus making G_C processable on-the-fly at query time.

Step 2: Selecting Edges of Subgraph G_C. The edge set E is defined as a sparse graph consisting of $|E| = 4 \cdot |V \backslash \{u, v\}| + 2$ double-directed edges which connect u to v and viceversa (2 edges), and every node in $V \backslash \{u, v\}$ to both u and v ($4 \cdot |V \backslash \{u, v\}|$ edges), and viceversa ($2k$ edges). The intuition is to use the nodes derived in Step 1 as *meaningful semantic bridges* for establishing the relatedness between u and v.

Step 3: Computing Weights of Edges in G_C. We define the weight of every edge $(x, y) \in E$ by deploying either the textual content of x's and y's pages, or their connectivity in G_C, or a proper combination of them (see Section 6.7). More precisely, the edge weight of $(n, m) \in E$ is computed with the following formula:

$$\text{weight}(n, m) = \frac{\text{rel}(n, m)}{\sum_{l \in V} \text{rel}(n, l)} \tag{1}$$

where $\text{rel}(n, m)$ is a relatedness score between $n \in V$ and $m \in V$, which will be chosen between Milne&Witten, Entity2Vec and DeepWalk, or their linear combination, selected as a result of the large experimental analysis conducted in Section 6.

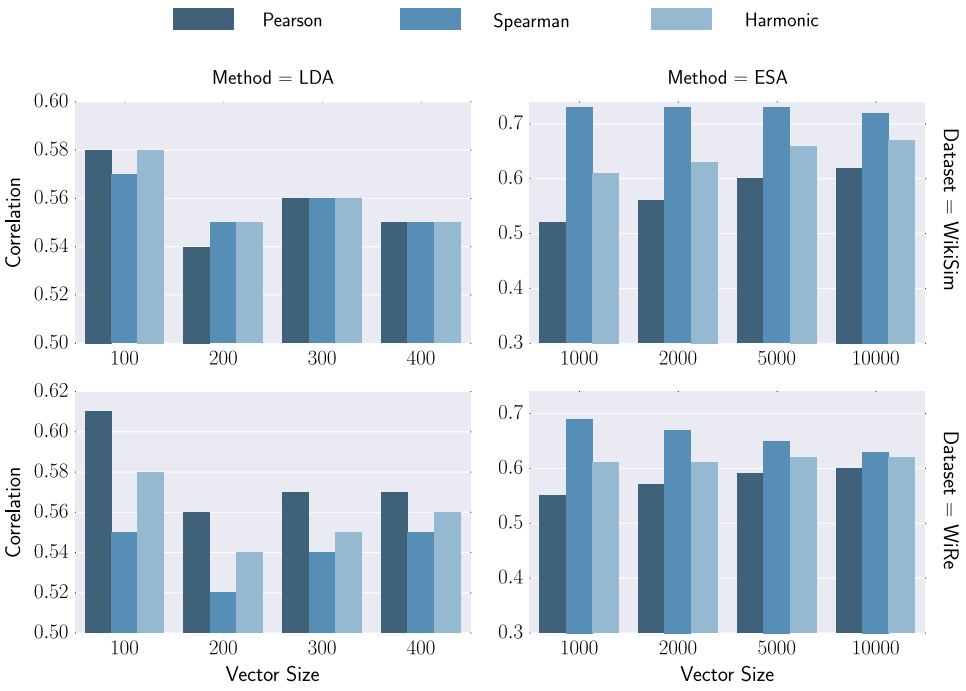


Fig. 5. Relatedness performance over WikiSim (top) and WiRe (bottom) datasets by ranging respectively the size of LDA and ESA vectors.

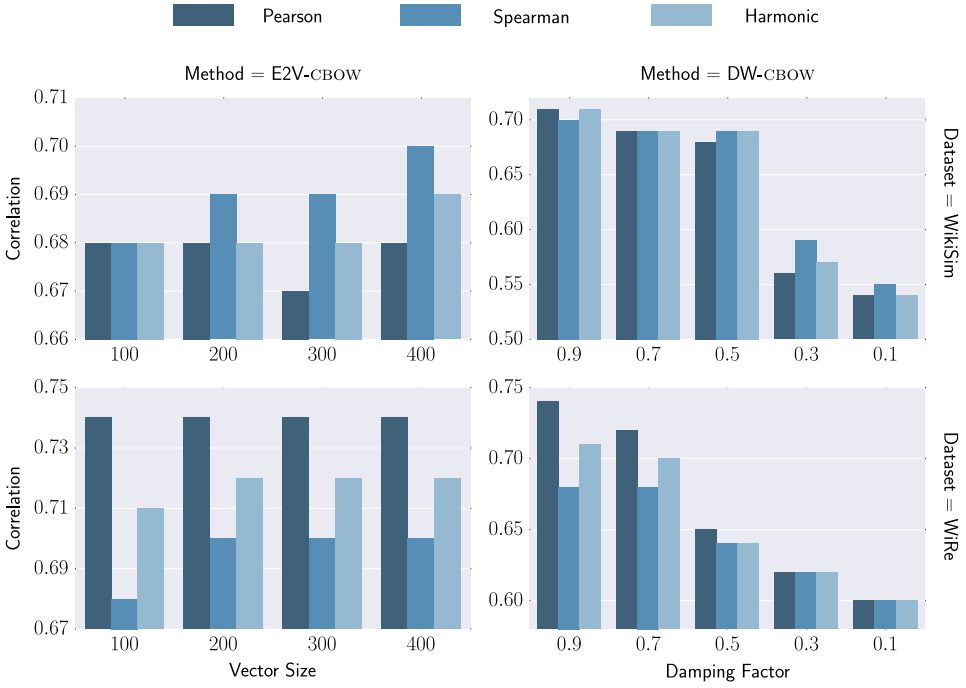


Fig. 6. Relatedness performance over WikiSim (top) and WiRe (bottom) datasets by ranging respectively the size of E2V-CBOW and DW-CBOW vectors.

5.2. Second stage: Computing relatedness

The relatedness score between u, v is derived by computing the CoSimRank [23] between these two nodes over the weighted subgraph G_C, given that this method is the one that best combines in a comprehensive way random walk approaches over weighted graphs. The key point here is that CoSimRank applied to G_C (as against the whole KG) is very fast because of the very small size of G_C. Specifically, CosimRank is executed by setting its personalized vector to the standard basis where all entries are 0 except for the 1-value associated to u (resp. v).

5.3. Discussion

Section 6 will investigate various combinations mentioned above for the Steps 1 and 3 (first stage), and show that these steps are particularly effective in estimating the relatedness between two query entities. We argue that this success can be attributed to several reasons. Edges in Wikipedia (or any typical KG) are only a noisy hint of relatedness since they may have been introduced by authors for different "goals": citation, elaboration, explanation, etc. Moreover, different communities of Wikipedia pages offer different "densities" of hyperlink annotations. These issues are

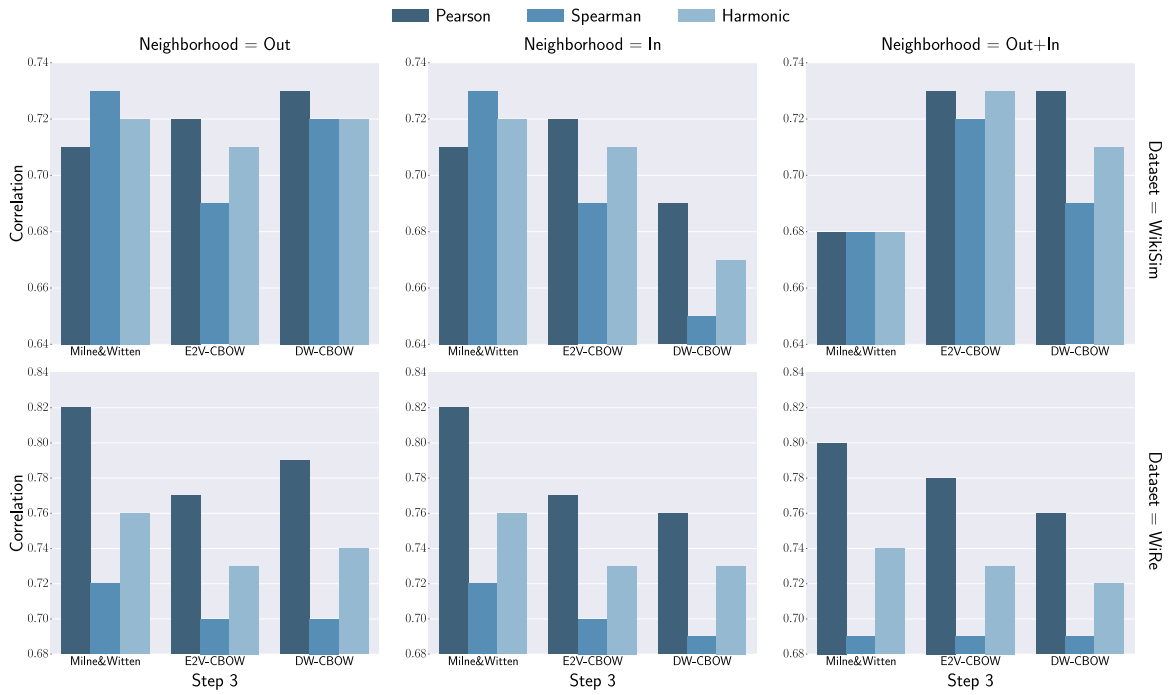


Fig. 7. Different configurations of our framework by using Milne&Witten for Step 1 and varying the choice of the neighborhood.

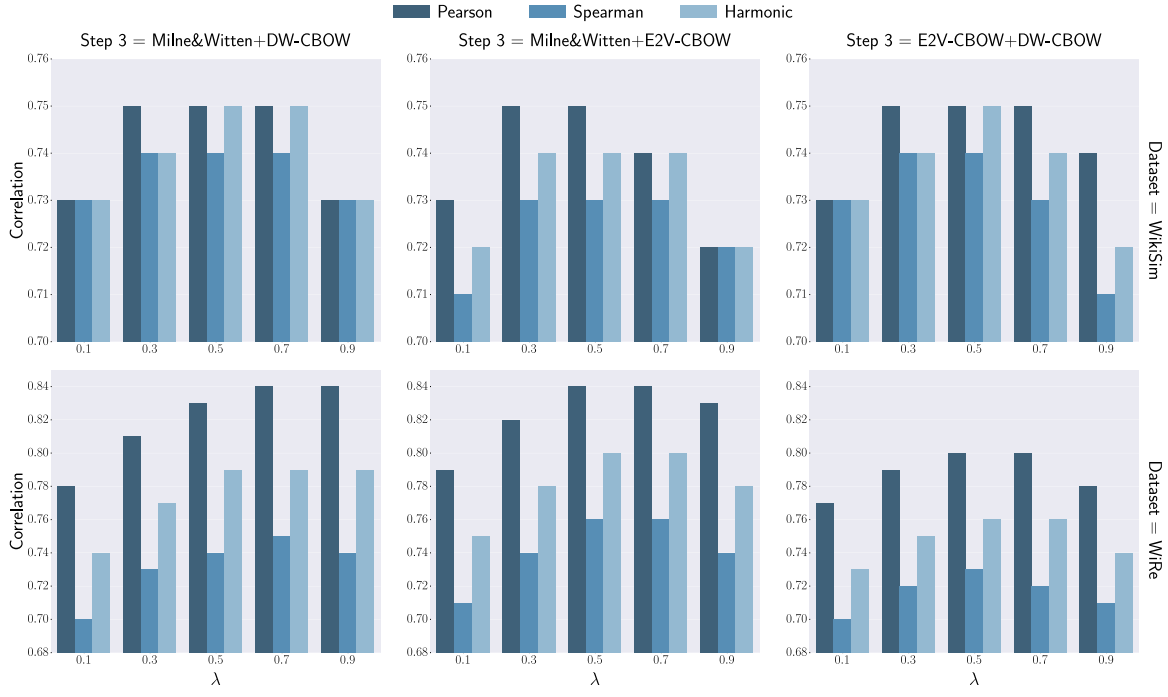


Fig. 8. Different configurations of our framework by using ESA for Step 1 and varying both λ and the linearly-combined methods of Step 3.

thus addressed by our framework through a proper selection of the nodes that constitute the subgraph G_C. Specifically, only the ones strongly related to the query nodes u and v in terms of textual content or neighbor structure constitute the potential "bridges" that are then used for the computation of their relatedness.

6. Experiments

In this section we first review the datasets used for our experimental analysis (i.e., WikiSim and WiRe). Then we describe configuration settings and implementations of the many algorithms we tested over these datasets. Next we present intrinsic evaluation results. Finally, we demonstrate extrinsic benefits among

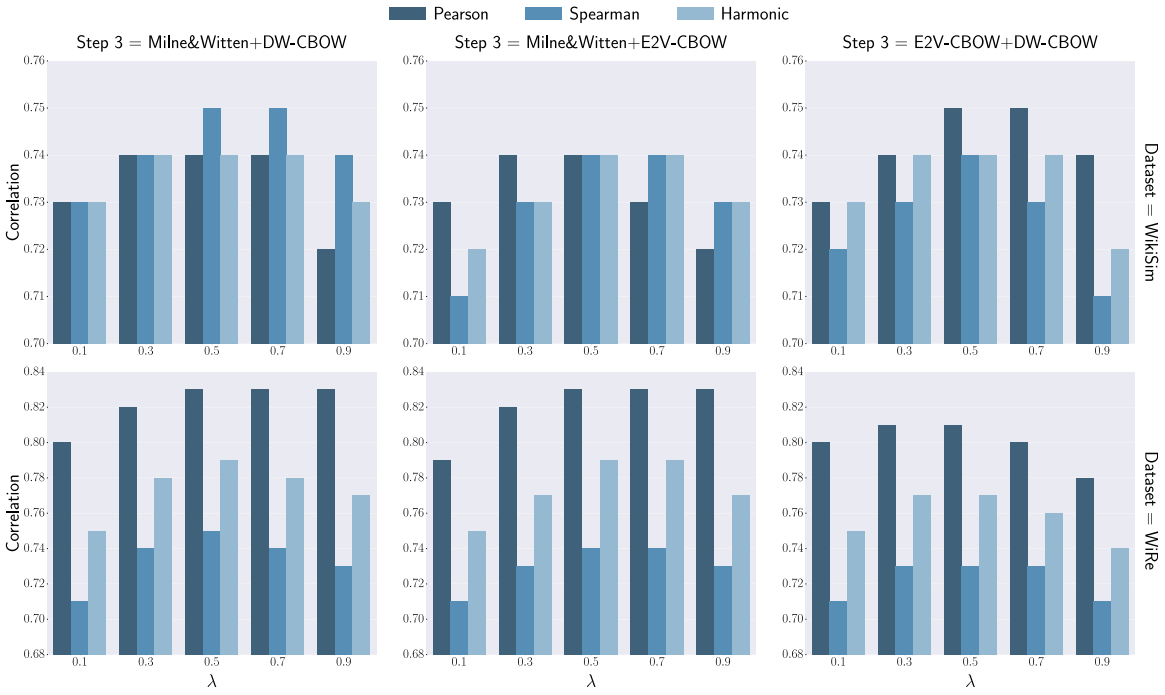


Fig. 9. Different configurations of our framework by using Milne&Witten for Step 1 and varying both λ and the linearly-combined methods of Step 3.

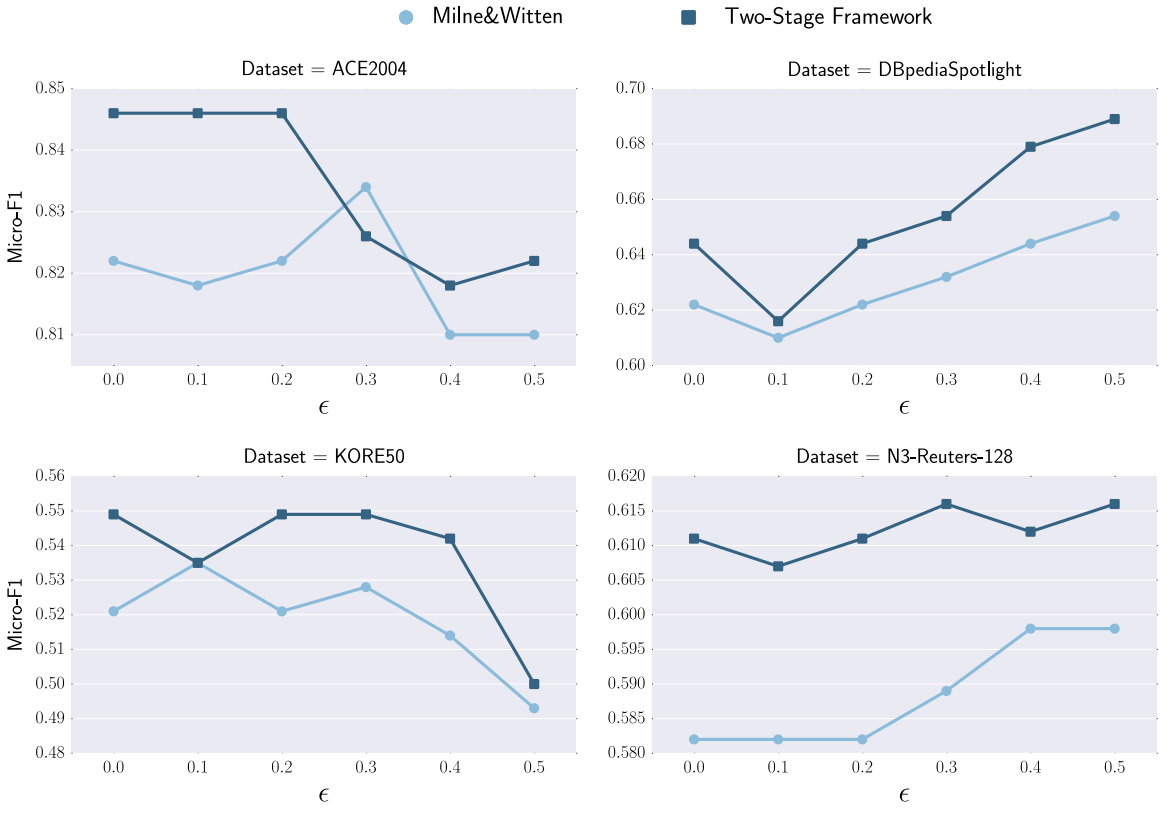


Fig. 10. Entity linking performance by varying the ε-score of TagMe's voting scheme which deploys Milne&Witten and our two-stage framework, respectively, over four distinct text corpora.

several applications in the domain of ranking entity pairs, entity linking, and synonym extraction. Datasets and all algorithms we experimented have been publicly released.[3]

6.1. Datasets

The first dataset we use is the one introduced by Ponza et al. [31], called WiRe. The other, WikiSim, is a version of the popular WordSim-353 dataset consisting of 353 word pairs, annotated with a relatedness score in [0, 10] via crowdsourcing, and then

[3] github.com/mponza/WikipediaRelatedness

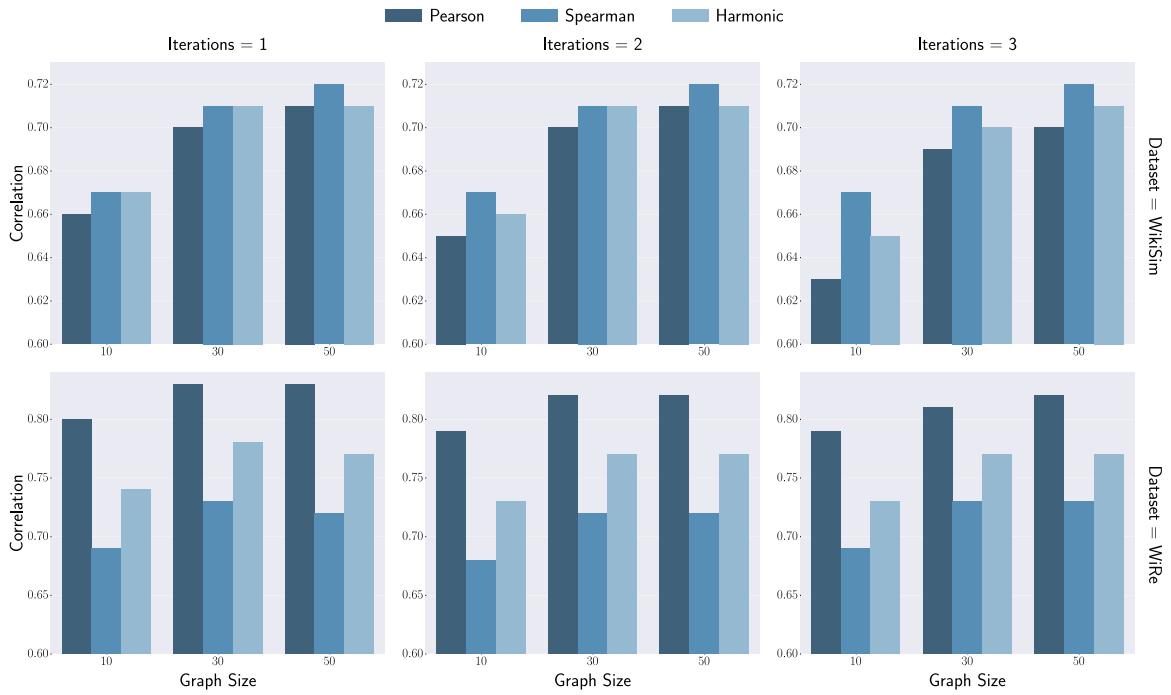


Fig. 11. Different configurations of our framework by instantiating Step 1 as ESA and Step 3 as Milne&Witten, and by varying the size of the graph and to the number of iterations.

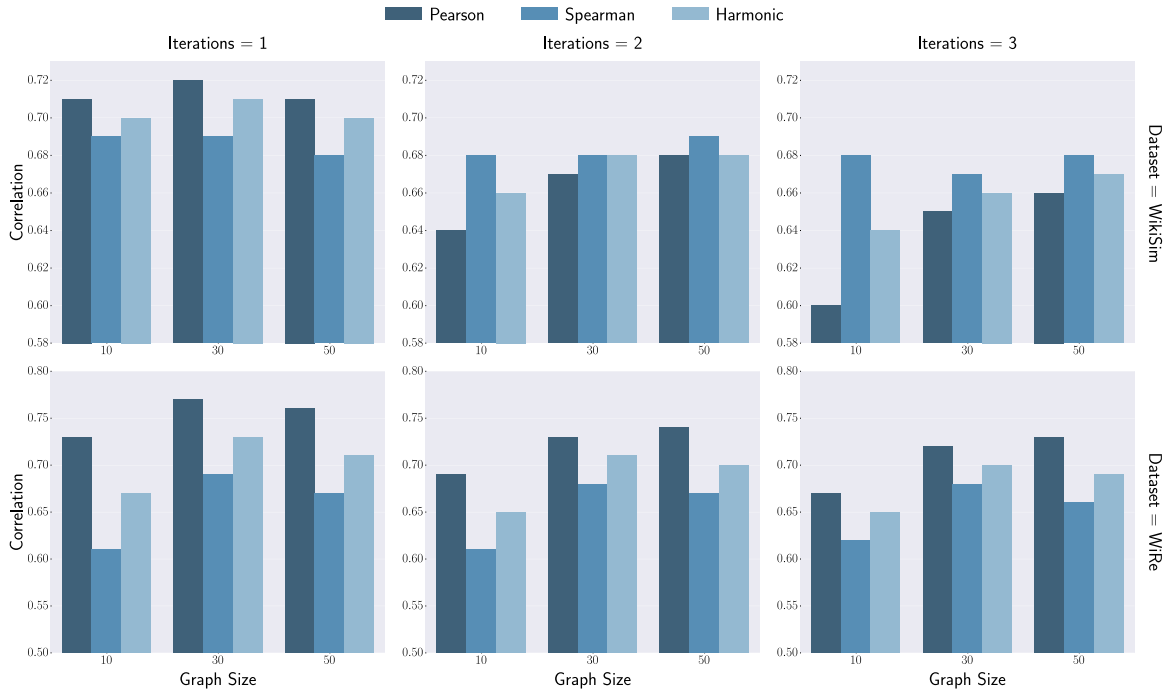


Fig. 12. Different configurations of our framework by instantiating Step 1 as ESA and Step 3 as E2V-CBOW, and by varying the size of the graph and to the number of iterations.

manually mapped into their corresponding Wikipedia entities by Milne and Witten [10]. We further filtered WikiSim (counts of discarded pairs are given in parentheses) by removing pairs whose entities are `null` (39), changing outdated Wikipedia IDs with current ones, removing disambiguation pages (33), removing duplicate pairs (4) and removing pairs (e, e) with relatedness score < 10 (9).

Fig. 3 reports some statistics about the two datasets. Relatedness scores in WiRe are integers, because they are the results of an agreement process between human assessors, whereas in WikiSim relatedness scores are reals obtained by averaging crowd-sourced scores. Nevertheless, WikiSim and WiRe have surprisingly similar relatedness score histograms, even though they were created using rather different processes. This lends confi-

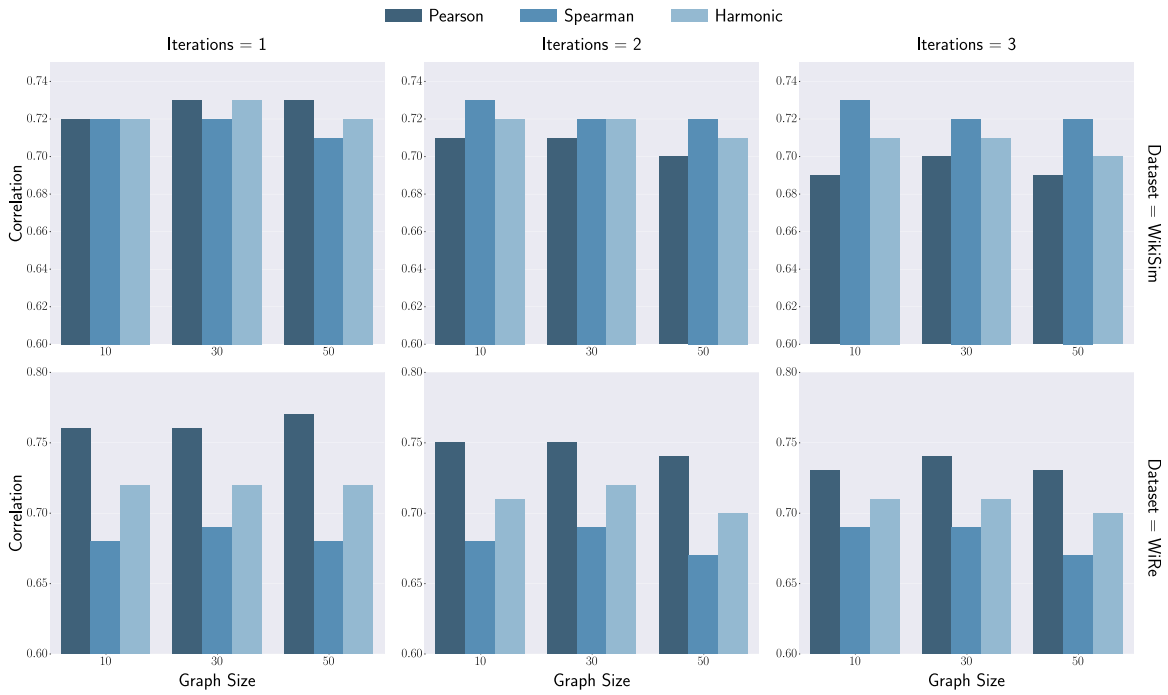


Fig. 13. Different configurations of our framework by instantiating Step 1 as ESA and Step 3 as DW-CBOW, and by varying the size of the graph and to the number of iterations.

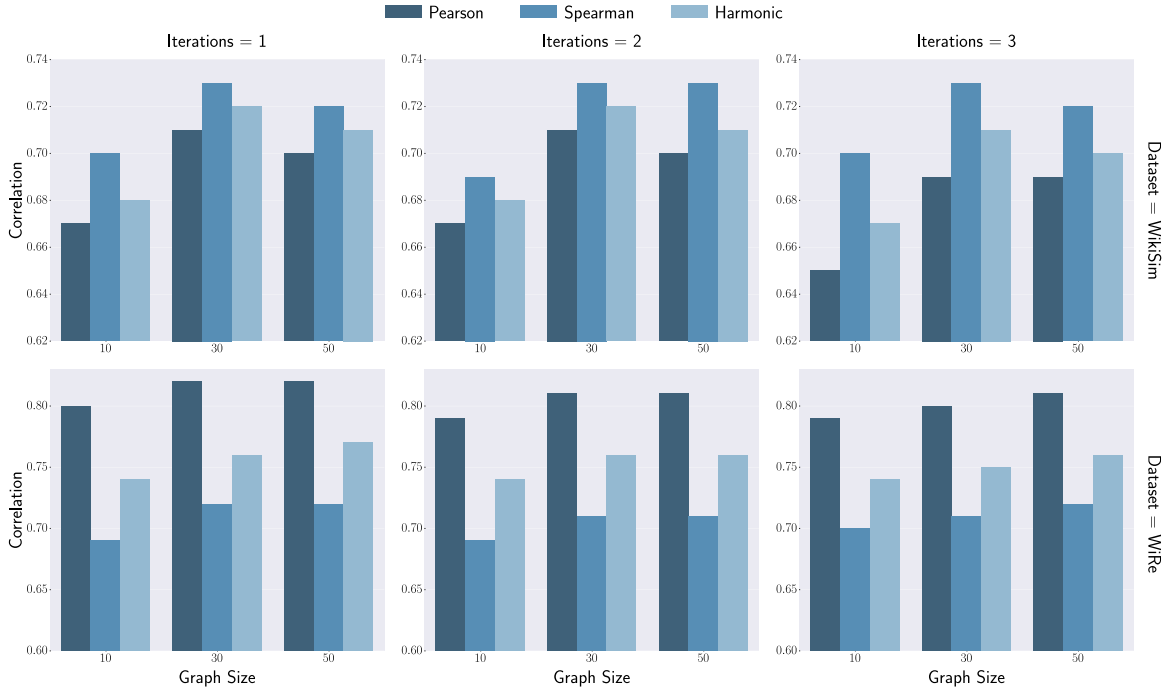


Fig. 14. Different configurations of our framework by instantiating Step 1 as Milne&Witten (computed over the out-neighborhood) and Step 3 as Milne&Witten, and by varying the size of the graph and to the number of iterations.

dence that accurate relatedness estimates can generalize across datasets.

Significantly (right column in Fig. 3) many entity pairs are similar (resp., dissimilar) yet far (resp., near) in the graph. Therefore, effective relatedness formulations *cannot assume* that KG distance between entities is a good predictor of their relatedness.

For the sake of completeness we mention that the datasets developed by Hoffart et al. [41], Ceccarelli et al. [18] offer a *ranking judgments* of a subset of entities related to some given ones. This is different from our dataset which offers direct *relatedness judgments* between entity pairs, which is a bigger challenge to entity relatedness measures.

6.2. System details

We give a few technical details about the implementation and the settings of the methods described in Section 3.

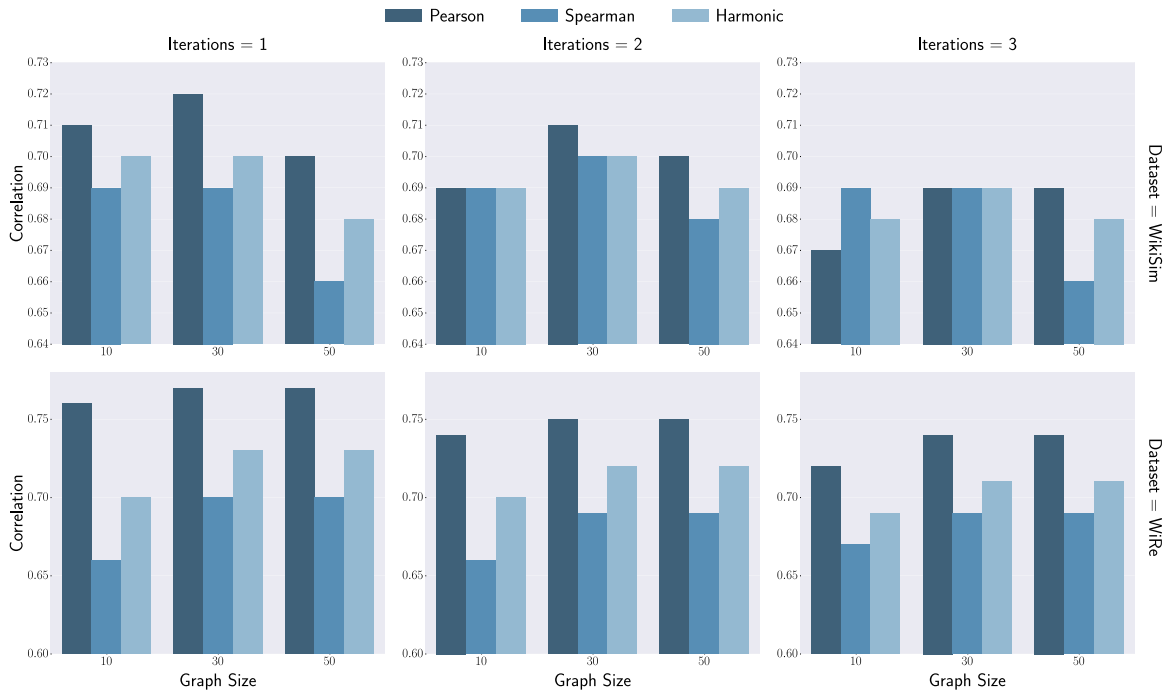


Fig. 15. Different configurations of our framework by instantiating Step 1 as Milne&Witten (computed over the out-neighborhood) and Step 3 as E2V-CBOW, and by varying the size of the graph and to the number of iterations.

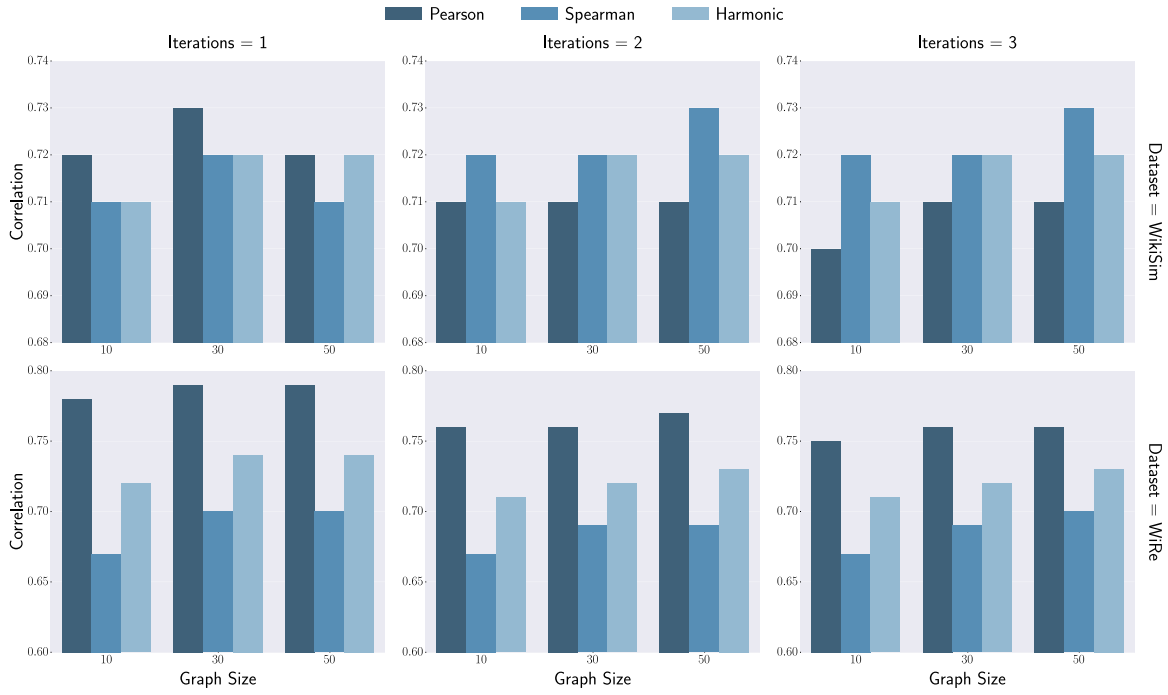


Fig. 16. Different configurations of our framework by instantiating Step 1 as Milne&Witten (computed over the out-neighborhood) and Step 3 as DW-CBOW, and by varying the size of the graph and to the number of iterations.

Wikipedia as Knowledge Graph. We used the Wikipedia dump of March 2016. The Wikipedia corpus was normalized by removing punctuation elements and lowercasing words. The Wikipedia graph was indexed with WebGraph [42] deriving the statistics reported in Fig. 4. CC means connected component. *Spid* indicates the shortest-path-index of dispersion that, being close to 0, shows that Wikipedia is closer to a social-network graph rather than to the Web graph. Important for the following algorithms is also the fact that the average distance is larger than 2, which is the maximum distance dealt with the well-known Milne&Witten's algorithm, and many pairs are concentrated around distances 3 and 4 (which are distances easily handled by our framework).

Implementation Specifics. We reimplemented ESA by Gabrilovich and Markovitch [9] and Ni et al. [1] because these software are not available. Wikipedia corpus was lemmatized with CoreNLP,[4]

[4] stanfordnlp.github.io/CoreNLP

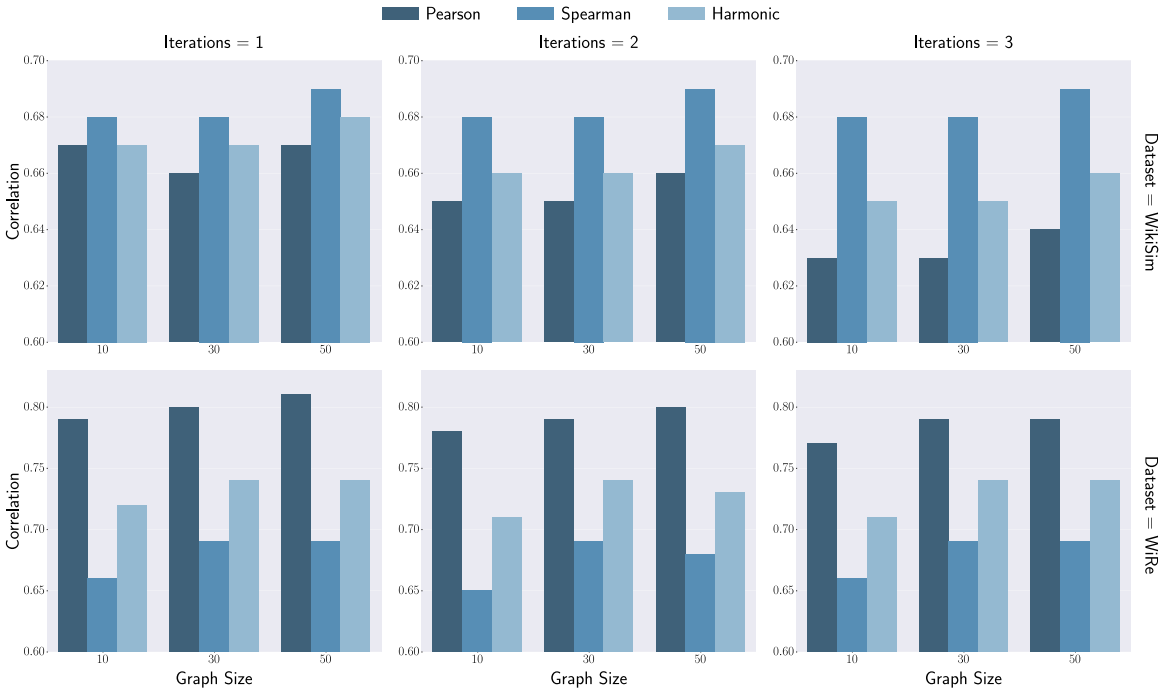


Fig. 17. Different configurations of our framework by instantiating Step 1 as E2V-CBOW and Step 3 as Milne&Witten, and by varying the size of the graph and to the number of iterations.

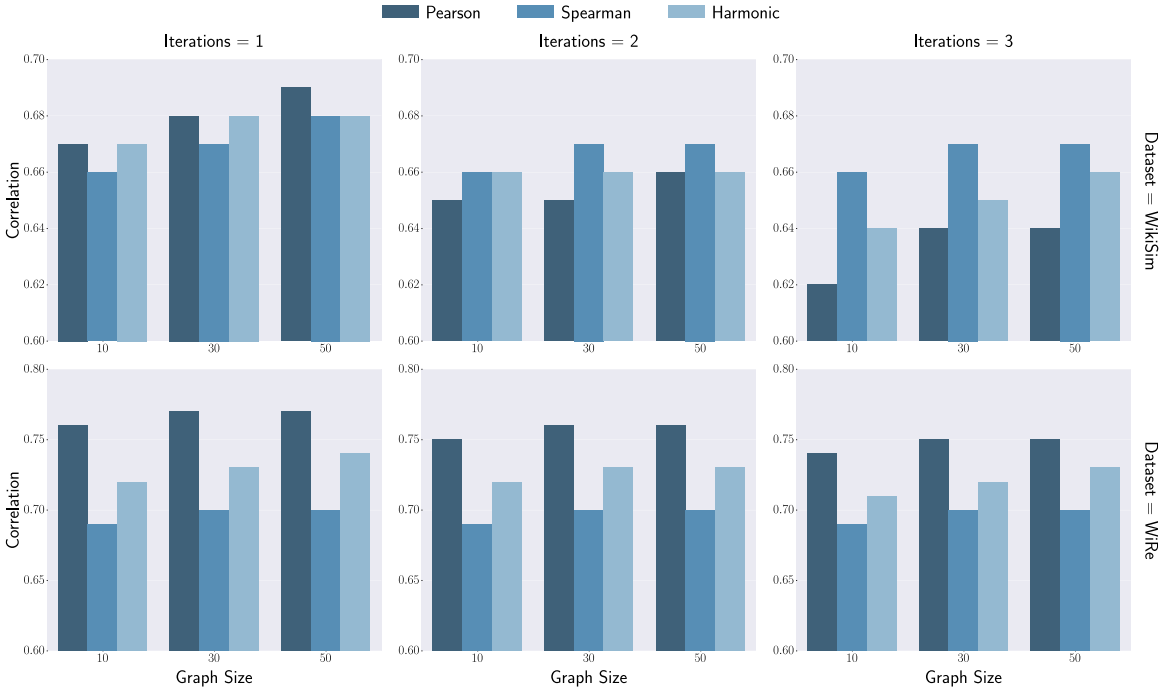


Fig. 18. Different configurations of our framework by instantiating both Step 1 and Step 3 as E2V-CBOW, and by varying the size of the graph and to the number of iterations.

stopwords removed, and indexed by Lucene.[5] Given an entity, its vector of concepts has been generated by using the Wikipedia page as query to the Lucene index with a BM25 ranking function.

For LDA we used its online stochastic variant [28] which has been shown to find topic models as good as, or better than its batch version. Both for LDA and Word2Vec we used the implementations available in Gensim.[6] To obtain Entity2Vec embeddings and LM probabilities, we replaced outbound hyperlinks to Wikipedia pages with a unique placeholder token [1], and processed this corpus using Word2Vec and BerkeleyLM [27] respectively. In DeepWalk we set the transition probability from u

5 lucene.apache.org

6 radimrehurek.com/gensim

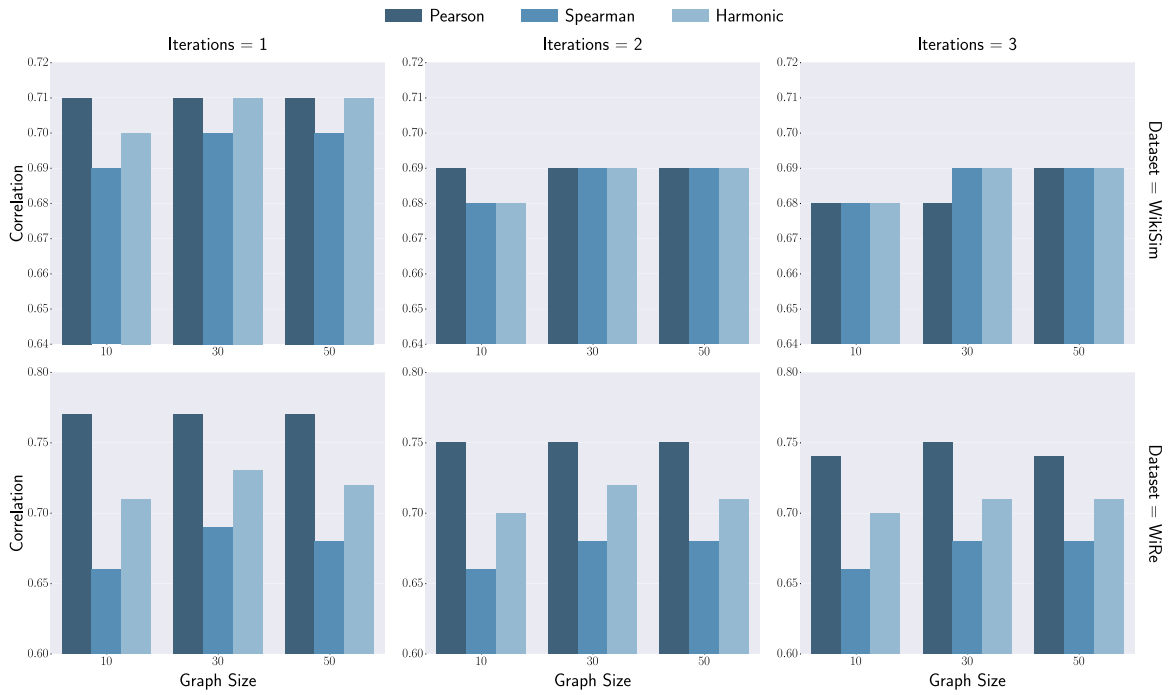


Fig. 19. Different configurations of our framework by instantiating Step 1 as E2V-CBOW and Step 3 as DW-CBOW, and by varying the size of the graph and to the number of iterations.

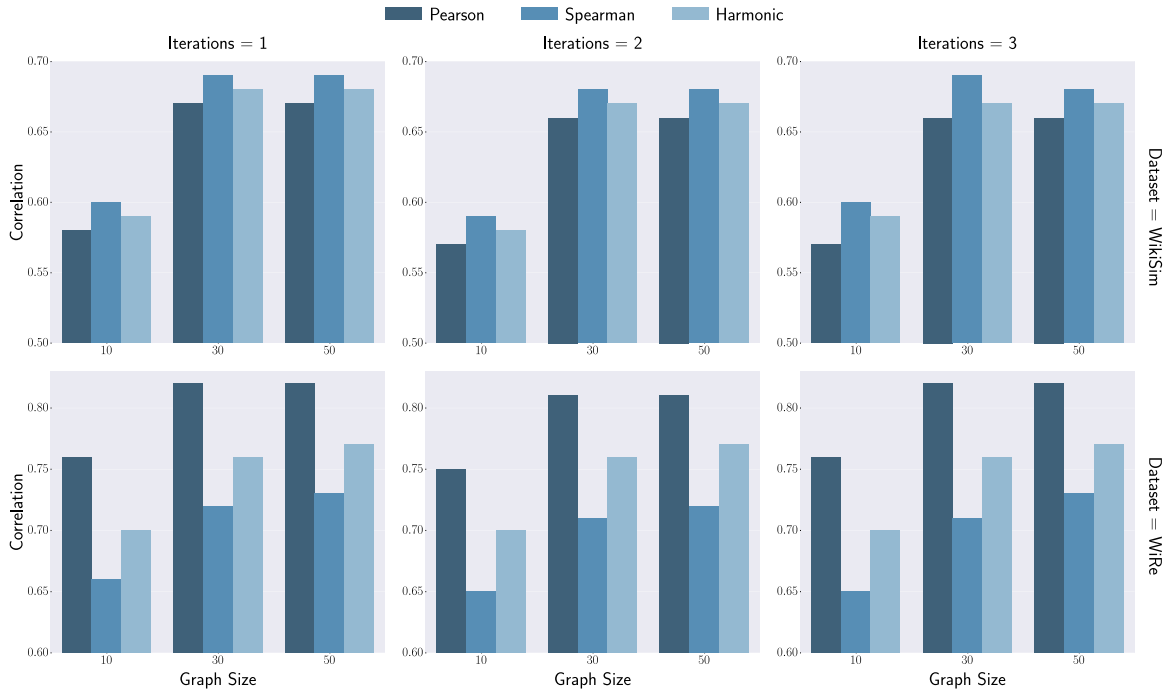


Fig. 20. Different configurations of our framework by instantiating Step 1 as DW-CBOW and Step 3 as Milne&Witten, and by varying the size of the graph and to the number of iterations.

to v as proportional to the frequency of a link to v within the textual description of u.

For the algorithms CoSimRank[7] and DeepWalk[8] we downloaded their official implementations and adapted them to work on our KG. We wrote WikiWalk from scratch, building upon the code of our ESA implementation.

6.3. Evaluation metrics

Evaluating relatedness scores against human judgment is complicated by the fact that prior work uses two different evaluation metrics: the **Pearson** correlation index [19,43], and the **Spearman** correlation index [9,37]. The Pearson index focuses on the difference between predicted-vs-correct relatedness *scores*. In contrast, the Spearman index focuses on the *ranking order* among

7 github.com/casaro/CoSimRank
8 github.com/phanein/deepwalk

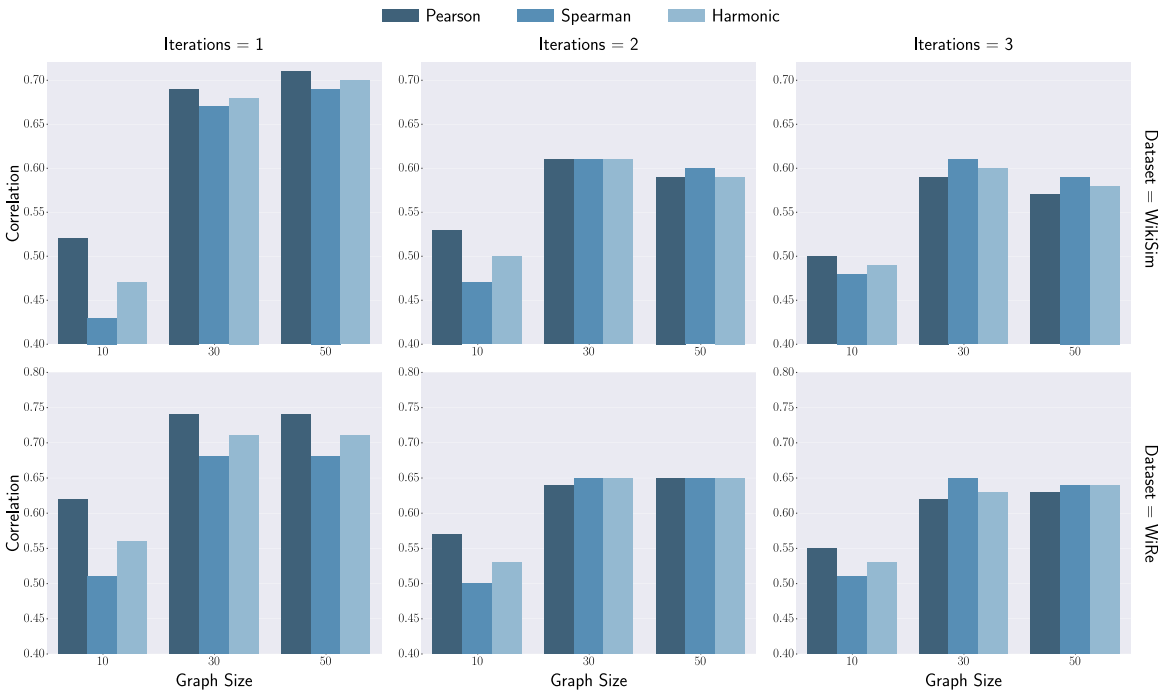


Fig. 21. Different configurations of our framework by instantiating Step 1 as DW-CBOW and Step 3 as E2V-CBOW, and by varying the size of the graph and to the number of iterations.

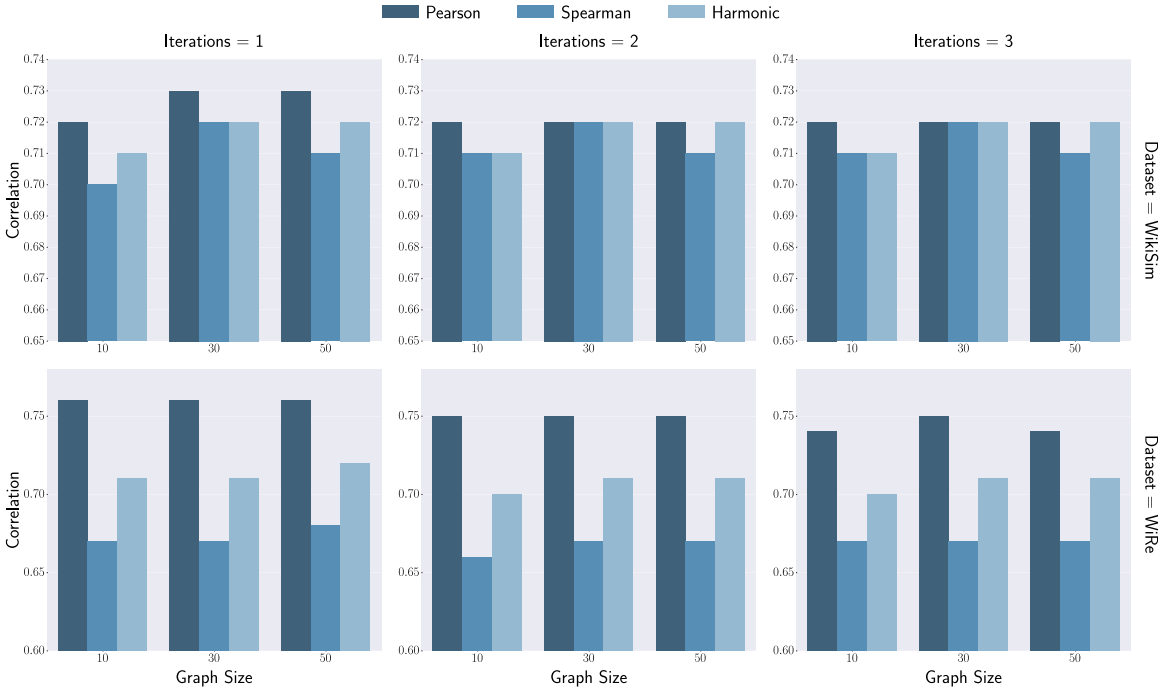


Fig. 22. Different configurations of our framework by instantiating both Step 1 and Step 3 as DW-CBOW, and by varying the size of the graph and to the number of iterations.

entity pairs as determined by the algorithm or by the human assessors.

Both indexes are meaningful because they capture different aspects which could be important to different extrinsic applications. Pearson's index is crucial when the *strength* of the entity relatedness needs to be correctly quantified, as it occurs in some entity linkers [1,5,14,16]. Spearman's index is crucial when the *correct ranking* among entity pairs based on their relatedness is required, as in other entity linkers [15,21].

Therefore, we follow Hassan and Mihalcea [44] and adopt three main measures: the two correlation indexes, i.e., Pearson and Spearman; plus their *harmonic mean*, to get one single score that is useful when the entity relatedness is interchangeably used for modeling coherence (strength) and candidate ordering (ranking) of entities (as in yet other entity linkers [8,45,46]). In addition, we use also the average of the harmonic means over the two datasets (marked AVG), just to have one unique score over all our experiments to compare methods. We measure

Algorithm 1 Algorithm implementing the step 1 of our two-stage framework, namely choosing the nodes of V for the subgraph $G_C = (V, E)$. TopRelNodes is a method (i.e., ESA, Entity2Vec, DeepWalk, or Milne&Witten) that returns the top-k most related nodes of a query node.

```
1: procedure ChooseNodes(u, v, k, TopRelNodes):
2:     V ← {u, v}
3:     R_u ← TopRelNodes(u, k)
4:     R_v ← TopRelNodes(v, k)
5:     i, j ← 0, 0
6:     while i < |R_u| and j < |R_v| and |V| < k do
7:         if i < |R_u| and |V| < k then
8:             V ← V ∪ R_u[i]
9:             i ← i + 1
10:        end if
11:        if j < |R_v| and |V| < k then
12:            V ← V ∪ R_v[j]
13:            j ← j + 1
14:        end if
15:    end while
16:    while i < |R_u| and |V| < k do
17:        V ← V ∪ R_u[i]
18:        i ← i + 1
19:    end while
20:    while j < |R_v| and |V| < k do
21:        V ← V ∪ R_v[j]
22:        j ← j + 1
23:    end while
24:    return V
25: end procedure
```

the statistical significance of our results with the methodology described by Radinsky et al. [12].

6.4. Comparative evaluation of building blocks

Toward selecting the best systems for our two-stage framework, we first compare all prior formulations and their various hyperparameters. We evaluate different lengths of ESA's concept vectors between [100, 10 000]. We try different lengths for the vectors built by LDA and Entity2Vec, between [100, 400]. We vary the damping factor of DeepWalk in [0.1, 0.9]. All results are reported in Figs. 5 and 6. Based on this study, for PPR + Cos, CoSimRank and WikiWalk, we set the damping factor to 0.8 and the number of iterations to 5. CoSimRank's weight decay is in {0.8, 0.9} while WikiWalk's teleportation vector is initialized with ESA's concept vectors of length 10 000.

The best performance settings are chosen to produce Table 5. For each method surveyed in Section 3, we show the four performance indexes. From the upper part of Table 5, we notice that the methods based on text show consistent performance across Spearman and Pearson indexes over both datasets.

Specifically, VSM, LM, and LDA achieve the lowest performance among the methods based on corpus text. The reason of such phenomenon is that these methods are too simple for managing the huge data-sparsity that words have in large-scale scenarios, as it occurs in our context with the Wikipedia corpus. Even applying standard preprocessing techniques – such as lowercasing, lemmatization, and stopwords removal – we were not able to make those approaches robust enough to overcome the sparsity of the dictionary at the hand. Conversely the more sophisticated methods, such as ESA and E2V, are more robust since they are able to model entities in a wider but denser space, thus resulting in more precise and stable results among all tested datasets.

From the lower part of Table 5, the methods based on KG structure show significantly different performance over the two indexes and the two datasets. Methods originally devised for node ranking achieve a very competitive Spearman's index at the cost of a worse Pearson's index (e.g., PPR + Cos and CoSimRank). Other methods show high and consistent performance on both datasets and indexes (e.g., Milne&Witten and DeepWalk variants). Milne&Witten, in spite of its simplicity, is among the best on our new WiRe dataset, with the best Harmonic coefficient. However, it lags in case of WikiSim. Even so, its extensive use in entity linkers [8] seems amply justified.

Among the methods based on the graph structure we can clearly identify a common pattern: the methods based on nodes' neighborhood perform poorer than all other methods in the table. This is clearly due to how their scores are computed. In fact, considering just the size of the intersection between two neighborhoods is a very weak signal for their relatedness, if this is not accompanied by the size of the intersected neighborhood. This is clearly shown by the poor performance achieved by the methods of Adamic–Adar, Bibliographic Coupling, Co-Citation, and Common Neighbors which use just the intersection size; slightly better results are obtained by normalized methods (such as Dice, Jaccard, and Milne&Witten).

As far as the random walk approaches are concerned (such as, PPR + Cos, CoSimRank, and WikiWalk), we observe that they are able to provide a good ranking of related entities (high Spearman metric), but they lack in properly quantifying the relatedness between single pairs. This is due to the fact that these methods take into account the whole Wikipedia graph and they compute multiple cosine-similarities between sparse vectors of millions of elements. So they tend to provide high scores to entities' which share a large number of nodes at short distances within their random walk, and very low scores to other entities. These limitations are usually overcome by the methods based on graph embeddings (i.e., DW-CBOW and DW-SkipGram), which seem to properly learn high quality dense vectors for most of the Wikipedia entities. Nevertheless, for few cases we noticed that the relatedness computed by DW is not accurate. This is probably due to how the context of entities is generated from the DW algorithm: randomly walking over the Wikipedia graph is an excellent strategy to make the learning algorithm scalable, but it leads to potentially miss different paths that could be useful for further improve the resulting embedding representation.

The top-performing methods over our datasets and correlation indexes are based on neural embeddings, such as E2V-CBOW and DW-CBOW. Although competitive harmonic means are obtained by Milne&Witten and ESA, their AVG performance is 3.5% and 6.5% lower than DW-CBOW respectively. PPR + Cos and CoSimRank obtain good Spearman's index but poor Pearson's index, which leads to low harmonic mean. This is not surprising since PPR + Cos and CoSimRank has been designed and used in contexts where only the ranking order between nodes in a graph was needed. Furthermore, since they involve the entire KG in their computation they are also very slow, we will no longer consider them in the remaining experiments.

6.5. Instantiating the two-stage framework

The experimental figures reported above guided us to choose four methods (i.e., ESA, E2V-CBOW, DW-CBOW and Milne&Witten) as component modules in our two-stage framework.

In Step 1, given query entities u and v, we extracted their top-k related entities by following one of the following three approaches:

Table 5
Relatedness performance of the methods surveyed in sections 3.1 and 3.2. For each correlation index, we color the first (in black), the second (in gray) and the third (in light gray) best performing method, respectively. The last column AVG reports the average of the harmonic means computed over the two datasets.

	Method	WikiSim			WiRe			AVG
		Pearson	Spearman	Harmonic	Pearson	Spearman	Harmonic	
Corpus Text	VSM	0.43	0.55	0.48	0.64	0.66	0.65	0.565
	LM	0.45	0.46	0.46	0.53	0.58	0.55	0.505
	ESA	0.62	0.72	0.67	0.60	0.63	0.62	0.645
	LDA	0.58	0.57	0.58	0.61	0.55	0.58	0.580
	E2V-CBOW	0.68	0.70	0.69	0.74	0.70	0.72	0.705
	E2V-SkipGram	0.67	0.66	0.66	0.74	0.69	0.71	0.685
Graph Structure	Bibliographic Coupling	0.34	0.58	0.43	0.51	0.63	0.57	0.50
	Co-Citation	0.19	0.62	0.29	0.20	0.47	0.28	0.285
	Common Neighbors	0.21	0.62	0.31	0.21	0.49	0.29	0.300
	Milne&Witten	0.62	0.65	0.63	0.77	0.69	0.72	0.675
	Jaccard	0.31	0.61	0.41	0.55	0.73	0.63	0.520
	Overlap	0.45	0.70	0.55	0.59	0.69	0.63	0.590
	Dice	0.35	0.67	0.46	0.49	0.70	0.58	0.520
	Adamic-Adar	0.20	0.63	0.31	0.19	0.49	0.28	0.295
	PPR+Cos	0.20	0.72	0.31	0.56	0.75	0.64	0.475
	CoSimRank	0.15	0.72	0.25	0.56	0.76	0.64	0.445
	WikiWalk	0.55	0.60	0.57	0.62	0.60	0.61	0.590
	DW-CBOW	0.71	0.70	0.71	0.74	0.68	0.71	0.710
	DW-SkipGram	0.67	0.70	0.69	0.73	0.67	0.70	0.695

- Top-k entities retrieved by ESA for u and v.
- Top-k entities having the highest cosine similarity with the embeddings of u, v computed by E2V-CBOW or DW-CBOW.
- Top-k nodes selected among the neighbors of u and v according to Milne&Witten score. This technique was experimented with the *out*-, *in*- and *out* + *in*-neighborhood.

For Step 2 of the first stage, we investigated three approaches to creating edges: either we created a *clique*, by connecting all pairs of nodes (thus generating $\Theta(k^2)$ edges), or we connected in a *bipartite* way the top-k nodes related to u with the top-k nodes related to v (again generating $\Theta(k^2)$ edges); or we *sparsified* the subgraph G_C by connecting u and v with only their top-k retrieved nodes (generating only $\Theta(k)$ edges). We report on this last approach because it achieved the best accuracy and speed.

In Step 3 of the first stage, we computed the weights of the $\Theta(k)$ edges created in Step 2 by determining the relatedness score between their endpoint nodes via three approaches: Milne&Witten, E2V-CBOW and DW-CBOW (we also investigated their Skip-gram-variants, but they performed worse than their CBOW counterparts).

For the second stage we used CoSimRank with damping factor and weight decay in $\{0.7, 0.8, 0.9\}$ and number of iterations 1–3. The results shown in the tables correspond to the best choices: damping factor and weight decay of 0.9 and one single iteration.

6.6. Two-stage results and analysis

The instantiation of Step 1 and Step 2 with different methods as well as varying the size of the subgraph and the number of iterations of our framework are used to generate Figs. 11 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22 that we report in Appendix. Since the number of results derived from the execution of these configurations, we comment here only the most relevant patterns spurred out from their collective analysis.

Generally speaking, our framework achieves the highest performances when only one CoSimRank iteration is computed. Increasing the number of iterations usually brings a degradation in the performance over both datasets and independently from the number of nodes of the subgraph. This behavior is clearly due to the way edges connect the nodes: in our sparse graph they connect the query nodes with all other nodes in the graph, thus allowing a random walker to "reach" all other nodes with only one single step (iteration). As far as the number of nodes used for the creation of the subgraph is concerned, the configurations with 30 and 50 nodes (and only one iteration) usually achieve very similar performances over both datasets and independently from the methods used for Step 1 and Step 3. Accordingly, from now on we fix the number of iterations to 1 and the size of the subgraph to 30.

From the experimental comparison just described above we can also state that the two-stage framework instantiated with Step 1 as Milne&Witten is one of the best performing configuration. Accordingly, we decide to analyze the performance of this configuration in several of its variations. Specifically, we vary both the neighborhood on which Milne&Witten is computed in Step 1 (i.e., out, in and both out + in) in order to generate Fig. 7. The configurations that use out- and in-neighborhood (first two columns) usually achieve higher performances than jointly using the out + in-neighborhood (last column). More precisely, when only the out-neighborhood is used in Step 1, performances are very similar to the one using the in-neighborhood, with the only exception when DW-CBOW is used as Step 3: here, performances are clearly higher than the ones showed with the in-neighborhood. This is caused by the fact that the out-neighborhood of a node in Wikipedia is usually small (with fewer than a hundred out-links on average), whereas the in-neighborhood can be very large (with thousands of incoming-links): selecting nodes from this last set can potentially generate

Table 6
Relatedness performance of our novel two-stage framework described in Section 5. Step 2 was configured by modeling the subgraph as a sparse graph (see text). The last line reports the best performance for each correlation index as showed in Table 5 (it was a black cell).

Step 1	Step 3	WikiSim			WiRe			AVG
		Pearson	Spearman	Harmonic	Pearson	Spearman	Harmonic	
ESA	Milne&Witten	0.70	0.71	0.71	0.83	0.73	0.78	0.745
	E2V-CBOW	0.72	0.69	0.71	0.77	0.70	0.73	0.720
	DW-CBOW	0.73	0.72	0.73	0.76	0.69	0.72	0.725
Milne&Witten	Milne&Witten	0.71	0.73	0.72	0.82	0.72	0.76	0.740
	E2V-CBOW	0.72	0.69	0.71	0.77	0.70	0.73	0.720
	DW-CBOW	0.73	0.72	0.72	0.79	0.70	0.74	0.730
E2V-CBOW	Milne&Witten	0.66	0.68	0.67	0.80	0.69	0.74	0.705
	E2V-CBOW	0.68	0.67	0.68	0.77	0.70	0.73	0.705
	DW-CBOW	0.71	0.70	0.71	0.77	0.69	0.73	0.720
DW-CBOW	Milne&Witten	0.67	0.69	0.68	0.82	0.72	0.76	0.720
	E2V-CBOW	0.70	0.67	0.68	0.74	0.68	0.71	0.695
	DW-CBOW	0.73	0.72	0.72	0.76	0.67	0.71	0.715
Best Results from Table 5		0.71	0.72	0.71	0.77	0.76	0.72	0.710

a graph with more noisy entities, not always mitigated by the weighting of edges performed in Step 3.

Table 6 reports the best results obtained for each one of the configurations described in the previous section. For the framework that uses Milne&Witten in Step 1, we report the configuration that uses the out-neighborhood since it achieves the highest performance in Fig. 7. Furthermore, to help the comparison against the best performance *for each individual correlation index* we report in the last line the best results achieved in Table 5. In the last column (AVG), we see that every system choice for Step 1 has at least one choice for Step 3 that outperforms the best results of Table 5, with $p < 0.05$ (the same p holds for all experiments in the next sections). Drilling down into individual correlation indexes (i.e., each column of Table 6), we notice that our two-stage framework improves all correlations indexes over all datasets except for the Spearman index over WiRe, for which CoSimRank still wins (0.76, i.e., +3% with respect to our best score). However, as mentioned in Section 6.4, CoSimRank is significantly slower because it works over the whole KG, while our approach works on the tiny subgraph G_C. Moreover, CoSimRank performs worse on other correlation indexes. We emphasize that the last row in Table 6, "Best Results from Table 5", does not represent any single method; different columns come from different methods in general. Therefore, the harmonic mean and AVG columns are upper bounds to the performance of each individual method of Table 5. Even compared to this non-constructive bound, our two-stage framework achieves *uniformly* the best performance on both datasets: +2% absolute over WikiSim and +6% absolute over WiRe.

Not surprisingly, given the results of the previous section, the configurations which achieve the top performance on WikiSim are the ones based on neural embeddings for Step 3 (namely, they use DW-CBOW), and the ones based on ESA or DW-CBOW for Step 1. More surprisingly, the configuration which achieves the top performance on WiRe is the one which does not rely on neural embeddings: namely ESA (Step 1 of first stage) and Milne&Witten (Step 3 of first stage). This configuration is more robust than the ones based on neural embeddings: it gets good performance also on WikiSim, whereas the others achieve slightly worse performance on WiRe dataset (thus gaining 2%–3% in the AVG over both datasets).

An interesting insight is that using Milne&Witten for Steps 1 and 3 amount to a form of *boosting* the influence diameter of their original approach. The structure of G_C makes our two-stage framework able to incorporate signals from paths of length 3 and 4 in the KG, instead of paths of length at most 2 as in Milne&Witten. This is crucial for distant node queries, which do indeed occur in our datasets (Fig. 3) as well as in applications.

6.7. Further improvements via combinations

In this section we take one further step and we investigate a linear combination of different configurations of Step 3 with the two best methods for Step 1 resulted from Table 6. We configured our two-stage framework with respectively ESA and Milne&Witten for Step 1, and then we modify the equation of Step 3 with the following formula:

$$\text{weight}(n, m) = \frac{\lambda \cdot \text{rel}_0(n, m) + (1 - \lambda) \cdot \text{rel}_1(n, m)}{\sum_{l \in V} \text{weight}(n, l)} \tag{2}$$

in order to linearly combine two different relatedness methods – i.e., rel_0 and rel_1 – for the weighting of our graph's edges $(n, m) \in E$. λ is chosen in 0.1–0.9.

Results of these configurations are shown in Figs. 8 and 9. Despite different methods linearly-combined for Step 3 and with different values of λ, both figures clearly show a common pattern: the highest performances are usually achieved when the two relatedness methods used for Step 3 are equally weighted. Accordingly, Table 7 reports the best results achieved by each combination obtained with $\lambda = 0.5$. Again, the last two lines are introduced for comparison and report the best performance *for each individual correlation index* in Tables 5 and 6, respectively. The improvement achieved by this combined scheme is again significant: with respect to the known methods of Table 5, its harmonic mean is +4% over WikiSim and +8% over WiRe (thus resulting in a +6% on AVG); with respect to our "uncombined" approach, the improvement in the harmonic mean is slight but consistent over the datasets.

The main reason why our framework achieves further improvements through these combinations is due to the fact that these methods complement each other. In particular, Milne&Witten is very accurate for entities that are close in the

graph, thus being able to further adjust the weights provided by using only DW or E2V. Furthermore, when farther entities are considered – i.e., distance greater than 2 – averaging the Milne&Witten score (i.e., which is zero) with the one provided by DW or E2V implicitly incorporates a distance signal in their relatedness computation, thus lowering those scores that are often not accurate and thus easily tend to be misleading.

Despite the best results are achieved by instantiating Step 1 with ESA, it actually comes with a larger running time than the framework when it deploys Milne&Witten for Step 1. Accordingly, in the next sections we focus the application of our framework configured with Milne&Witten as Step 1 and either Milne&Witten or the linear combination between Milne&Witten and DW-CBOW as Step 3. As we will show, not only can one of these two computationally lightweight configurations of our framework be easily adapted and deployed in other domains and applications, but they will also bring improvements in terms of either accuracy or computational time.

Comparison against E5 System. Table 8 reports the results of the two best configurations of our framework against the E5 system, a recent proposal of Zeng et al. [30] appeared after the publication of our paper [31]. Our framework achieves very competitive results, with performances that are only slightly lower than E5. On the other hand, authors of E5 reported in [30] that "computing the relatedness score given two entities can be achieved within *seconds*" by using their system, which is clearly a high running time, especially if a number of relatedness scores need to be computed on-the-fly, such as it happens in the entity linking domain.

Conversely, as Section 6.9 will show, our framework runs in milliseconds thus offering an appealing trade-off between efficacy (being just −1/2% less accurate than E5) and efficiency (being at least two order of magnitude faster).

6.8. Applications

In this section we show the effectiveness and flexibility of our two-stage framework through its application in different contexts. More precisely, we investigate the application of our proposal in three different domains: (1) *ranking entity pairs*, the task which concerns the proper ordering of a set of candidate entities with respect to a given seed entity, (2) *entity linking*, the task which concerns the annotation of sequence of words with unambiguous entities, and (3) *synonyms extraction*, the task which concerns the retrieval of one or more words that are synonyms of a given one.

Ranking Entity Pairs. In this section we evaluate our novel two-stage framework in the domain of ranking pairs of Wikipedia entities [41]. Different from the original settings on which our framework has been designed (i.e., general-purpose relatedness computation), the problem of ranking entity pairs concerns only the *ordering* of a set of candidate entities (from the most to the lowest relevant) with respect to a given seed entity.

For the evaluation, we use the KORE dataset [41], constituted by a total of 21 seed entities grouped in 5 different topics (i.e., IT Companies, Hollywood Celebrities, TV Series, Video Games, and Chuck Norris). Each seed entity is associated with a set of 20 candidate entities, whose gold-standard ranking was created by human assessors via crowdsourcing [41]. As baselines we use KORE (Original), KORE (LSH-F), and KORE (LSH-G) [41], a set of unsupervised corpus-based techniques proposed by Hoffart et al. [41] for the ranking of entity pairs and which are currently the state-of-the-art on the KORE dataset. Specifically, the ranking of entities is performed by these methods through an algorithm based on the weighted overlap of keyphrases vectors that are associated in an off-line phase to the KG's entities. KORE (original) implements this algorithm exactly, whereas KORE (LSH-F) and KORE (LSH-G) speed-up the computation with an approximated algorithm based on different settings of LSH [47,48].

Unfortunately, 28 entities of the KORE dataset are not present in our KG, so we remove them and compute the Spearman correlation of the original KORE techniques[9] on the remaining instances. In more detail, we removed 4 entities belonging to IT Companies, 6 entities to Hollywood Celebrities, 5 to TV Series, 11 to Video Games and 2 to Chuck Norris. Given a seed entity e_s and its set of associated candidate entities C_{e_s}, our two-stage framework ranks every $e_c \in C_{e_s}$ accordingly to the relatedness score computed between e_s and $e_c \in C_{e_s}$.

Results are reported in Table 9. The two different configurations of our two-stage framework (last two rows) obtain very good performance among all different entity's topics. More precisely, they achieve the highest two scores in Hollywood Celebrities and TV Series topics, with improvements of +5.4% and +18.8% with respect to the third-highest methods KORE (LSH-G). Our framework also achieves the second-highest performance among IT Companies and Chuck Norris, with a small difference of −2.1% and −2.5% with respect to KORE (Original) and KORE (LSH-F). The lowest performance (where our method arrives third and fourth) is achieved on the Video Games topic, with a difference with respect to the first one of −18.8% and −18.5%. This is not surprising since almost half of the entities of the KORE dataset we removed belong to this topic. The lack of Video Games entities in our KG makes the two-stage framework unable to create meaningful subgraphs for entities belonging to this topic, by eventually producing inaccurate relatedness scores. Nevertheless, our framework achieves the overall best first and second average performance (last two columns of Table 9), both per topic and among all seed entities, with improvements up to +2.9% and +4.4% with respect to KORE (Original).

Evaluation on TagMe. We supplement the evaluations above with an extrinsic evaluation on entity linking, also called named entity disambiguation, a standard NLP task. In the public domain, TagMe [16] is among the most popular, open-source[10] and well-known entity linkers. It has served over 400 million annotations to date. The disambiguation algorithm in TagMe relies on a voting scheme which assigns a weight to each candidate entity by combining Milne&Witten's score with the commonness of the candidate entity (entity prior probability). Finally, the entity assigned to a significant text span is the most common entity among the ones that exceed the score $(1 - \varepsilon)\ \gamma$, where ε is a tuned tolerance and γ is the maximum voting score reported by a candidate entity for that text fragment.

We replaced the relatedness method used in TagMe with our two-stage framework, configured as Milne&Witten for both Steps 1 and 3 of the first stage. For the sake of completeness, we mention that we also applied the configuration of our framework by using the linear combination between Milne&Witten and DW-CBOW, but it actually resulted in lower performance. Fig. 10 shows the performance of TagMe by varying its parameter ε in the commonly-used range [0, 0.5] (i.e., it chooses the best voted entity, $\varepsilon = 0$, or considers commonness among the entities scoring half of the highest vote, $\varepsilon = 0.5$). On four diverse datasets [8], our relatedness measure not only improves TagMe, but also makes it more insensitive to choices of ε.

Evaluation on Synonym Extraction. In this last section we set up one more experiment that investigates the application of

[9] We thank Johannes Hoffart for providing us the outputs of KORE methods.
[10] github.com/gammaliu/tagme

Table 7
Comparing the two best methods for Step 1 with a linear combination of different relatedness methods for Step 3. λ is set to 0.5. .

Step 1	Step 3	WikiSim			WiRe			AVG
		Pearson	Spearman	Harmonic	Pearson	Spearman	Harmonic	
ESA	Milne&Witten + E2V-CBOW	0.75	0.73	0.74	0.84	0.76	0.80	0.770
	Milne&Witten + DW-CBOW	0.75	0.74	0.75	0.84	0.75	0.79	0.770
	E2V-CBOW + DW-CBOW	0.75	0.74	0.75	0.80	0.73	0.76	0.755
Milne&Witten	Milne&Witten + E2V-CBOW	0.74	0.74	0.74	0.83	0.74	0.79	0.765
	Milne&Witten + DW-CBOW	0.74	0.75	0.74	0.83	0.75	0.79	0.765
	E2V-CBOW + DW-CBOW	0.75	0.74	0.74	0.81	0.73	0.77	0.755
Best Results from Table 5		0.71	0.72	0.71	0.77	0.76	0.72	0.710
Best Results from Table 6		0.73	0.72	0.73	0.83	0.73	0.78	0.745

Table 8
Comparison between the best configurations of our framework from Table 7 and E5 system [30]. Despite our framework achieves performance slightly lower than the system proposed in [30], it comes with a significantly lower running time (few milliseconds) than E5 (seconds).

Step 1	Step 3	WikiSim			WiRe			AVG
		Pearson	Spearman	Harmonic	Pearson	Spearman	Harmonic	
ESA	Milne&Witten + DW-CBOW	0.75	0.74	0.75	0.84	0.75	0.79	0.770
Milne&Witten	Milne&Witten + DW-CBOW	0.74	0.75	0.74	0.83	0.75	0.79	0.765
	E5 System [29]	0.76	0.75	0.76	0.85	0.77	0.81	0.785

Table 9
Relatedness performance on the task of ranking entity pairs over the KORE dataset [41]. The first five columns distinguish the performance among different entity's topics, whereas the last two columns summarize them by their average. The last two rows report the two-stage framework instantiated with Milne&Witten for Step 1 and two different configurations for Step 3.

Method	IT Companies	Hollywood Celebrities	TV Series	Video Games	Chuck Norris	AVG (per Topic)	AVG (All Entities)
KORE (Original)	0.750	0.634	0.479	0.765	0.587	0.643	0.655
KORE (LSH-F)	0.185	0.512	0.386	0.457	0.705	0.449	0.400
KORE (LSH-G)	0.601	0.642	0.508	0.718	0.587	0.611	0.616
Step 3 = Milne&Witten	0.668	0.696	0.674	0.577	0.640	0.651	0.677
Step 3 = Milne&Witten + DW-CBOW	0.729	0.676	0.696	0.580	0.680	0.672	0.699

Table 10
Results on the application of the two-stage framework in the domain of synonyms extraction and experimented on the TS68 dataset. k is the size of the subgraph of our two-stage framework (set to 20, i.e. value that achieved the highest performance), d is the average degree of each node of the graph (i.e., 66) and m is the number of synonym pairs in the testbed (i.e., 68). All rows, with the exception of the last one, are from [23].

Method	P@1	MRR	Time complexity
PPR + Cos	20.7	32.0	$\mathcal{O}(mn^2)$
SimRank	25.0	**37.0**	$\mathcal{O}(n^3)$
CoSimRank	25.5	**37.0**	$\mathcal{O}(mn^2)$
Typed CoSimRank	23.5	**37.0**	$\mathcal{O}(mn^2)$
Two-Stage Framework	**26.6**	33.0	$\mathcal{O}(\boldsymbol{kdmn})$

our two-stage framework (configured with Step 1 and 3 with Milne&Witten relatedness) on the domain of *synonym extraction*: the task for which CoSimRank [23] was originally designed and experimented. In this context, our goal is to compare our framework against other unsupervised state-of-the-art graph-based methods for synonym extraction and see if our technique can bring further improvements either in quality or efficiency terms. Specifically, since our framework runs CoSimRank on a properly reduced and weighted subgraph created from a larger one, with this experiment we aim at showing how much the performance of our technique could differ with respect to the original CoSimRank executed over the larger graph.

For a fair comparison, we run our framework on the same graph of words deployed by Rothe and Schütze [23], which consists of 30 945 nodes and 2 045 411 edges, and over the same benchmark, namely TS68, a dataset published by Minkov and Cohen [49] consisting of 68 pairs of synonyms. We easily adapted our two-stage framework to retrieve the synonyms of a word as follows. Given a query node (word) u, its synonyms are retrieved by ranking all nodes in the graph with respect to their relatedness score with u. Results are reported in Table 10. P1 and MRR are two evaluation measures that respectively compute the precision in the first position and the Mean Reciprocal Rank [50].

Our framework achieves competitive results, with performances that are better or near to the state-of-the-art, but with the great advantage of having a time complexity of one order of magnitude lower. We also mention that our approach does not take advantage of the edge weights (i.e., word co-occurrences computed on a large corpus) provided by the larger graph of words. These weights are actually exploited by all other methods reported in Table 10, thus further penalizing our results that take into account only the un-weighted edge connections.

6.9. Optimizations and efficiency

The efficiency of our framework heavily depends on the efficient construction of the weighted subgraph G_C. In this section, we show that, by carefully optimizing a few steps, we are able to determine the relatedness of two entities sufficiently fast at query time.

Table 11
Un/compressed space occupancy (MBytes) of the Wikipedia graph and DW-CBOW's embeddings.

	Wikipedia graph	DW-CBOW embeddings
Uncompressed	605	4418
Compressed	209	236

Table 12
Running time performance (milliseconds) of our two-stage framework configured as in Table 7.

		DW-CBOW			
		WikiSim		WiRe	
		Uncomp.	Comp.	Uncomp.	Comp.
Milne & Witten	Uncomp.	0.4	1.7	0.7	0.9
	Comp.	1.9	2.4	3.3	4.1

Milne&Witten is much more efficient than ESA in retrieving the top-k related nodes which may be precomputed at preprocessing time and stored in compressed form (see next), whereas ESA needs us to execute a possibly expensive query on Lucene. However, the accuracy of Step 1 configured as Milne&Witten is close to that using ESA. Therefore, we analyze the scalability of our two-stage framework instantiated with Milne&Witten for Step 1 and Milne&Witten+DW-CBOW for Step 3.

We precompute some data to speed-up the construction of the subgraph G_C as follows:

Step 1: We need to retrieve the top-k out-neighbors of u and v according to the Milne&Witten relatedness score. So we precompute and store for every node in the Wikipedia graph its top-k nodes, this takes kn integers (node-IDs).

Step 3: The subgraph G_C needs to compute the edge weights between u (resp. v) and the nodes extracted in Step 1. Since the edge weights are a combination of Milne&Witten's score and DW-CBOW's score, we need to store at every node in the Wikipedia graph its adjacency list (for the former score); and an s-size embedding of floating-point numbers (for the latter score).

Overall this takes kn integers and kn floating-point numbers[11] and it can be reduced via compression. We applied Elias–Fano codes from WebGraph [42] to compress the KG, and we used FEL [6] to compress the embeddings. Table 11 reports the results of this experiment. The compressed graph is three times smaller than its uncompressed version. The compressed DW-CBOW embeddings are almost twenty times smaller than the uncompressed ones.

Given this precomputed information the relatedness query between two nodes u, v is executed by applying CoSimRank over the graph G_C for just one single iteration (see the Second Stage of Section 5). Looking to the algorithmic structure of CoSimRank one can deduct that this computation corresponds to the *cos*-similarity between two vectors of size k, which contain as values the weights of the edges connecting u, v to the k nodes extracted by Step 1. So the query takes $\mathcal{O}(k \cdot (d+s))$ time because the weight of the edge (x, y) needs to process the adjacency list of x, y for deriving the Milne&Witten's score (this has average length d), and the s-size embeddings of x, y. Recalling that $k = 30$, $s = 100$ and $d = 20$ in Wikipedia, it is easy to conclude that the relatedness query is very fast. Table 12 details the average query time with un/compressed graph and embeddings (over 10 different runs executed on an AMD Opteron 6238 clocked at 2600MHz, with 128GB of RAM, running Linux 3.13). The deployment of compressed Wikipedia graph for Milne&Witten induces a 5 times slower query with a 3× space-saving. The use of compressed embeddings is very convenient, because, despite introducing a very small-time overhead, the space is two orders of magnitude less.

Overall the compressed solution can fit all the needed information in 60% of the space required by Wikipedia graph uncompressed, and perform relatedness queries in less than 5 ms.

[11] Since node-IDs are not consecutive in Wikipedia one needs to index them for allowing constant access time. This additional space is not accounted for in Table 11, thus giving further advantage to the uncompressed space solution.

7. Conclusion and future work

In this paper, we studied the problem of computing relatedness between entities in Wikipedia. We re-implemented and used a number of techniques that have been previously proposed in recent literature in different domains but that have never been applied before over this task. On the top of them, we proposed a new framework that improves the state-of-the-art results through the deployment of two stages that respectively involve the creation of a small and weighted subgraph around two query entities and then the calculation of their relatedness through computation over this graph. We experimental studied a number of configurations that can be used for the instantiation of these two stages as well as we showed the effectiveness of our novel technique over three different domains: entity linking, ranking entity pairs, and synonym extraction.

Our work also leaves open some issues that deserve attention in the future. The two-stage framework does not currently take into account labels or weights that can be present in the graph at the hand and that could actually be used to further improve its performance. Moreover, our technique currently works only on input pairs, while an extension and application upon multiple inputs (e.g., set of words, like texts) could actually bring a significant impact many other domains, such as document similarity, the core task on which most of the relatedness techniques have been developed so far.

Acknowledgments

We warmly thank Francesco Piccinno for fruitful discussions and help in constructing the WiRe dataset and for kindly providing us access to several resources he generated during his PhD. Part of the work of the first two authors has been supported by a Bloomberg Data Science Research Grant (2017), and by the EU grant for the Research Infrastructure "SoBigData: Social Mining & Big Data Ecosystem" (INFRAIA-1-2014-2015, agreement #654024). Part of the work of the third author has been supported by an Amazon and IBM research grant.

Appendix

See Figs. 11–22.

References

[1] Y. Ni, Q.K. Xu, F. Cao, Y. Mass, D. Sheinwald, H.J. Zhu, S.S. Cao, Semantic documents relatedness using concept graph representation, in: Proceedings of WSDM, 2016.
[2] I. Bordino, G.D.F. Morales, I. Weber, F. Bonchi, From machu picchu to "rafting the urubamba river": anticipating information needs via the entity-query graph, in: Proceedings of WSDM, 2013.
[3] U. Scaiella, P. Ferragina, A. Marino, M. Ciaramita, Topical clustering of search results, in: Proceedings of WSDM, 2012.
[4] P. Cifariello, P. Ferragina, M. Ponza, Wiser: a semantic approach for expert finding in academia based on entity linking, Inform. Syst. (2019).
[5] M. Cornolti, P. Ferragina, M. Ciaramita, S. Rüd, H. Schütze, A piggyback system for joint entity mention detection and linking in web queries, in: Proceedings of WWW, 2016.
[6] R. Blanco, G. Ottaviano, E. Meij, Fast and space-efficient entity linking for queries, in: Proceedings of WSDM, 2015.

[7] E. Meij, W. Weerkamp, M. De Rijke, Adding semantics to microblog s, in: Proceedings of WSDM, 2012.
[8] R. Usbeck, M. Röder, A.-C. Ngonga Ngomo, C. Baron, A. Both, M. Brümmer, D. Ceccarelli, M. Cornolti, D. Cherix, B. Eickmann, et al., GERBIL: general entity annotator benchmarking framework, in: Proceedings of WWW, 2015.
[9] E. Gabrilovich, S. Markovitch, Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis, in: Proceedings of IJCAI, 2007.
[10] D.N. Milne, I.H. Witten, An effective, low-cost measure of semantic relatedness obtained from Wikipedia links, in: Proceedings of AAAI, 2008.
[11] E. Agirre, E. Alfonseca, K.B. Hall, J. Kravalova, M. Pasca, A. Soroa, A study on similarity and relatedness using distributional and WordNet-based approaches, in: Proceedings of NAACL-HLT, 2009.
[12] K. Radinsky, E. Agichtein, E. Gabrilovich, S. Markovitch, A word at a time: computing word relatedness using temporal semantic analysis, in: Proceedings WWW, 2011.
[13] J. Dunietz, D. Gillick, A new entity salience task with millions of training examples, in: Proceedings of EACL, 2014.
[14] S. Kulkarni, A.V. Singh, G. Ramakrishnan, S. Chakrabarti, Collective annotation of Wikipedia entities in web text, in: Proceedings of SIGKDD, 2009.
[15] S. Cucerzan, Large-scale named entity disambiguation based on Wikipedia data, in: Proceedings EMNLP-CoNLL, 2007.
[16] P. Ferragina, U. Scaiella, Fast and accurate annotation of short texts with wikipedia pages, IEEE Softw. (2012).
[17] W. Shen, J. Wang, J. Han, Entity linking with a knowledge base: Issues, techniques, and solutions, IEEE Trans. Knowl. Data Eng. (2015).
[18] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, S. Trani, Learning relatedness measures for entity linking, in: Proceedings of CIKM, 2013.
[19] M. Strube, S.P. Ponzetto, WikiRelate! Computing semantic relatedness using Wikipedia, in: Proceedings of AAAI, 2006.
[20] T. Nguyen, T. Tran, W. Nejdl, A Trio Neural Model for Dynamic Entity Relatedness Ranking, in: Proceedings of EMNLP-CoNLL, 2018.
[21] F. Piccinno, P. Ferragina, From TagME to WAT: a new entity annotator, in: International Workshop on ERD, SIGIR, 2014.
[22] T.H. Haveliwala, Topic-sensitive pagerank, in: Proceedings of WWW, 2002.
[23] S. Rothe, H. Schütze, Cosimrank: A flexible & efficient graph-theoretic similarity measure, in: Proceedings of ACL, 2014.
[24] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, LINE: Large-scale information network embedding, in: Proceedings of WWW, 2015.
[25] B. Perozzi, R. Al-Rfou', S. Skiena, DeepWalk: online learning of social representations, in: Proceedings of SIGKDD, 2014.
[26] D. Liben-Nowell, J. Kleinberg, The link-prediction problem for social networks, in: Proceedings of JAIST, 2007.
[27] A. Pauls, D. Klein, Faster and smaller n-gram language models, in: Proceedings of ACL, 2011.
[28] M.D. Hoffman, D.M. Blei, F.R. Bach, Online learning for latent Dirichlet allocation, in: Proceedings of NIPS, 2010.
[29] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of NIPS, 2013.
[30] W. Zeng, J. Tang, X. Zhao, Measuring entity relatedness via entity and text joint embedding, Neural Process. Lett. (2018).
[31] M. Ponza, P. Ferragina, S. Chakrabarti, A two-stage framework for computing entity relatedness in Wikipedia, in: Proceedings of CIKM, 2017.
[32] T. Maehara, et al., Computing Personalized PageRank quickly by exploiting graph structures, in: Proceedings of VLDB, 2014.
[33] G. Jeh, J. Widom, SimRank: a measure of structural-context similarity, in: Proceedings of SIGKDD, 2002.
[34] D. Fogaras, B. Rácz, Scaling link-based similarity search, in: Proceedings of WWW, 2005.
[35] I. Hulpuş, N. Prangnawarat, C. Hayes, Path-based semantic relatedness on linked data and its use to word and entity disambiguation, in: Proceedings of ISWC, 2015.
[36] P. Torres-Tramón, C. Hayes, A random walk model for entity relatedness, in: European Knowledge Acquisition Workshop, 2018.
[37] E. Yeh, et al., WikiWalk: Random walks on Wikipedia for semantic relatedness, in: In International Workshop on GMNLP, 2009.
[38] A.P. F. Fouss, M. Saerens, A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation, in: Proceedings of WI, 2005.
[39] S. Banerjee, A. Roy, Linear Algebra and Matrix Analysis for Statistics, Chapman and Hall/CRC, 2014.
[40] O. Levy, Y. Goldberg, Neural word embedding as implicit matrix factorization, in: Proceedings of NIPS, 2014.
[41] J. Hoffart, S. Seufert, D.B. Nguyen, M. Theobald, G. Weikum, KORE: keyphrase overlap relatedness for entity disambiguation, in: Proceedings of CIKM, 2012.
[42] P. Boldi, S. Vigna, The WebGraph framework I: Compression techniques, in: Proceedings of WWW, 2004.
[43] M. Mohler, R. Mihalcea, Text-to-text semantic similarity for automatic short answer grading, in: Proceedings of EACL, 2009.
[44] S. Hassan, R. Mihalcea, Semantic relatedness using salient semantic analysis, in: Proceedings of AAAI, 2011.
[45] S. Zwicklbauer, C. Seifert, M. Granitzer, Robust and collective entity disambiguation through semantic embeddings, in: Proceedings of SIGIR, 2016.
[46] O.-E. Ganea, M. Ganea, A. Lucchi, C. Eickhoff, T. Hofmann, Probabilistic bag-of-hyperlinks model for entity linking, in: Proceedings of WWW, 2016.
[47] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, in: Proceedings of STOC, 1998.
[48] A. Gionis, P. Indyk, R. Motwani, et al., Similarity search in high dimensions via hashing, in: Proceedings of VLDB, 1999.
[49] E. Minkov, W.W. Cohen, Graph based similarity measures for synonym extraction from parsed text, in: Proceedings of TextGraphs, 2012.
[50] D.C. Manning, P. Raghavan, H. Schacetzel, Introduction to information retrieval, Nat. Lang. Eng. (2008).