Weighted-Distance Sliding Windows and Cooccurrence Graphs for Supporting Entity-Relationship Discovery in Unstructured Text

Paolo Fantozzi, Luigi Laura, Umberto Nanni

***Abstract*—The problem of Entity relation discovery in structured data, a well covered topic in literature, consists in searching within unstructured sources (typically, text) in order to find connections among entities. These can be a whole dictionary, or a specific collection of named items. In many cases machine learning and/or text mining techniques are used for this goal. These approaches might be unfeasible in computationally challenging problems, such as processing massive data streams.

A faster approach consists in collecting the cooccurrences of any two words (entities) in order to create a graph of relations - a cooccurrence graph. Indeed each cooccurrence highlights some grade of semantic correlation between the words because it is more common to have related words close each other than having them in the opposite sides of the text.

Some authors have used sliding windows for such problem: they count all the occurrences within a sliding windows running over the whole text. In this paper we generalise such technique, coming up to a Weighted-Distance Sliding Window, where each occurrence of two named items within the window is accounted with a weight depending on the distance between items: a closer distance implies a stronger evidence of a relationship. We develop an experiment in order to support this intuition, by applying this technique to a data set consisting in the text of the Bible, split into verses.

***Keywords*—Cooccurrence graph, entity relation graph, unstructured text, weighted distance.

I. Introduction

THE size of the information created every day is about 2˙500 Petabytes[1]. Most of these data are unstructured (text, images, etc.). So, one of the biggest challenge, of these years, is the identification of underlying structures in the unstructured data. In this case, the managing of unstructured textual data, though if well studied, is far from being fully covered.

Since the size of data about entities contained in repositories, such as Wikipedia and Freebase, increases, the focus of research about unstructured text is shifted to the entity relation discovery. This paper attempts to address this challenge, focusing specifically on entity relation discovery on a corpus composed by unstructured text.

Entity relation discovery for structured data is a well covered topic in literature, specially where the entities are already identified [2]- [4]. In many cases machine learning and text mining techniques are used for this goal [5], [6].

To the best of our knowledge, there are no techniques that exploit the distance between words in unstructured text to build a graph of the entities. Most of the techniques uses just the binary indication of two entities in the same document to assign a relation between them [7]. The cooccurrence of two words (entities) can be exploited to create a graph of relations, but the information about the distance of the words should be preserved in order to maintain the semantic content of the grammar text construction. Indeed implicitly this information highlight the grade of correlation between the words because it's more common to have related words close each other than having them in the opposite sides of the text.

II. Related Work

The cooccurrence graphs are tools used often in literature, in many different forms. The simplest form is the unweighted graph where an edge is present if two words appear in the same document and it is not present if there is no document which both appear in. Sometimes thresholds on the number of documents are used to reduce the noise as in [8].

Some works focused on clustering a cooccurrence graph, where the edges are just binary indicator of the existence of a document containing both words. In this case the noise introduced by the statistical behaviour of the language is reduced by the clustering action as in [9].

Other works used the weighted version of the cooccurrence graph built on a corpus with entities as nodes. In this case the weights on the edges are equal to the number of the documents containing both entities. Then the graph is analysed as a social network as in [7]. One more interesting work based on the weighted cooccurrence graph is [10] with the goal of measure the abstractness of a word.

One important use of the cooccurrence graph is in the field of word sense disambiguation. Reference [11] uses unweighted cooccurrence graphs built using a sliding window, if two words are in the same window then there will be an edge between them. Reference [12] extends this approach by using a weighted graph based on the frequency of cooccurrence of two words in all the paragraphs.

Reference [13] uses weighted cooccurrence graphs with sliding windows of different sizes to extract keywords and sentences from the document to summarise it. In their results the techniques reach a state of the art precision and f-score

Paolo Fantozzi is with the Centre for Transport and Logistics (CTL), Sapienza University of Rome, Rome, Italy (e-mail: paolo.fantozzi@uniroma1.it).

Luigi Laura and Umberto Nanni are with the Centre for Transport and Logistics (CTL) and Department of Computer, Control, and Management Engineering, Sapienza University of Rome, Rome, Italy (e-mail: luigi.laura@uniroma1.it, umberto.nanni@uniroma1.it).

[1] 2.5 quintillion of bytes as reported IBM in [1]

but they highlight a lower recall with respect to the other supervised techniques.

Reference [14] proposes an alternative to *tf-idf* score using cooccurrence of words. Instead of using a graph, it uses a squared matrix containing, in each cell, how many sentences contains both words. Then, using frequency, clustering and other techniques, it computes a score for each word.

Reference [15] uses a weighted cooccurrence graph to estimate provinces geographical distance. The weights are equal to the number of web pages which the provinces occur together in. The graph is represented in form of a matrix and then its columns are compared to compute a distance. In similar way [16] uses a cooccurrence weighted graph based on number of cooccurrences in web pages, in order to infer geographical hierarchies of places' names.

Reference [17] searches for visual words (recurrent structures) in images. These visual words are used as entities in a cooccurrence graph and each image is used as a document. The graphs are, then, compared to re-identified the same person in two different images.

A more complex way to build a semantic graph, in order to cluster by topic, is shown in [18] after the algorithm proposed in [19] called ICAN method. While in [20] the cooccurrence is replaced with a dimensionality reduction of the text by using a random indexing technique, in order to cluster topics.

III. Applications

A weighted graph of words (or entities) can be exploited in many fields of application. It is possible to use it to discover changes in human-written ontologies. In this case a simple comparing between edges can highlight both new or previously undiscovered relations between entities. For instance in the field of medicine, where many handwritten ontologies are still alive, it could be used to analyse new articles to help the experts to find new relations.

Another field of application could be the identifying of aliases of entities [21]. In this case, a cooccurrence graph, could indicate two entities which aliases one of the other simply seeing to their neighbourhoods. Indeed, two aliases of the same entities, are probably used with the same set of words, so to identify two aliases it is sufficient to identify two nodes with the same distribution of weights among the same set of nodes.

One more field of application is the automated data cleaning in text [22]. Many texts contain references to different entities using the same abbreviation (Donald L. White and Donald E. White, both of whom are referred to as D. White, as remarked in [22]). Using a cooccurrence graph could help to disambiguate the references comparing the other words in the document to compare their distribution with the relations in the graph.

Reference [23] uses entity relations from unstructured text to analyse and link entities in emails' text. The relations are, then, compared with other public sources (for instance LinkedIn) to build a network given from the union of unstructured and structured data.

IV. Our Proposal

Our proposal is to build a cooccurrence graph with weights based on the distance between words in text. Each node is a word and each edge weight is the score assigned to the cooccurrence of the two words connected. A high score could be obtained in different ways: two words often close each other in different documents, two distant words very often in very many documents.

The resulting graph is a representation of the relations between words in text which keeps the information about the distance between words. This information is bounded to the semantic of the phrases, and so it is more complete than the counting for how many documents contain the same pair of words.

The importance of weighting the distances is even more evident if we consider a "long" document. In this case the number of unweighted cooccurrences is very high, but the relevance of these is questionable. Even using a window (unweighted version), there is a trade off between a wide and a short size. A short one may miss many cooccurrences, and a long one may fail to distinguish between dramatically different "closeness" degrees.

A. Algorithm

Without loss of generality, we assume that the document in input is already tokenized by words. For each word, the scores between that word and all the others in the window are computed. Using a structure based on hash, the sum of the scores for each pair of words is cached, to keep reading and writing constant in time.

Algorithm 1 Weighted cooccurrence Graph

```
function WeiCoocGraph(words, weights, threshold)
    graph ← new Graph()
    scores ← new Dictionary()
    for $i = 0;\ i < words.size();\ i++;$ do
        $left \leftarrow words[i]$
        $maxj \leftarrow \min(i + weights.size() + 1,\ words.size())$
        for $j = i + 1;\ j < maxj;\ j++;$ do
            $right \leftarrow words[j]$
            $l \leftarrow \min(left,\ right)$
            $r \leftarrow \max(left,\ right)$
            $k \leftarrow j - i - 1$
            $weight \leftarrow weights[k]$
            $scores[l,\ r] \leftarrow scores[l,\ r] + weight$
            if $l \neq r$ & $scores[l,\ r] > threshold$ then
                $graph[l,\ r] \leftarrow scores[l,\ r]$
            end if
        end for
    end for
    return graph
end function
```

Algorithm 1 (Weighted Cooccorrence Graph) shown above has cost of $O(k \cdot n)$ in time, where n is the size of the scanned text and k is the size of the sliding window. The required space is of $O(k \cdot m)$, where $m = O(min\{kn, w^2\})$ is the number of relationship collected while running the algorithm, and w is the number of entities. In practical terms, we get a running time which is essentially linear in the size of scanned text (the input), and a space which is linear in the size of the total

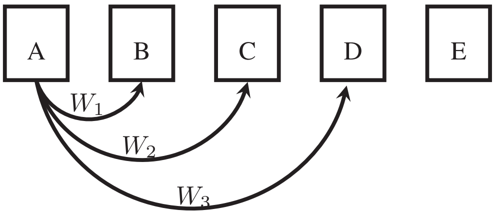

Fig. 1 A window of entities. Let us consider the entities A, B, C, D, E, and a window of size 4. Cooccurrence (A, B) has weight W_1, (A, B) has weight W_2, (A, B) has weight W_3. Since the size of the window is 4, then A and E are considered non-cooccurrent

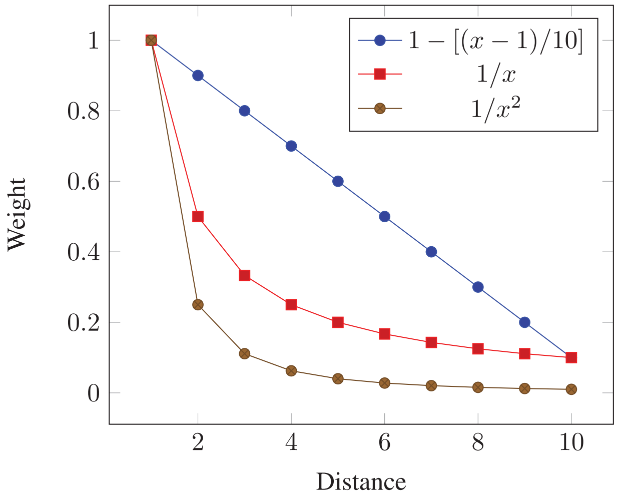

Fig. 2 Weight function examples. Weighted functions should be (strictly) descending. Each field of application could have a weight function that best fits the semantic in the text

number of retrieved cooccurrences. The resulting cooccurrency graph (the output) might have a smaller size due a threshold greater than zero.

B. Window and Weights

The weights are placed in an array starting with index 1. Each cell contains the weight defined for the distance equal to its index. The weights should be (strictly) decreasing to maintain the information about the distance between words.

The document is analysed by using a sliding window of words. Inside the window, then, the weights are computed on the basis of the first left word with respect to the others as showed in Fig. 1.

The size of this window and the weight for each position are parameters of the algorithm. The bigger is the window and the more connected is the resulting graph. Indeed, a window of size 10 means that all the words that are distant more than 10 spots will be ignored while computing the score.

The function used to compute the weights assigns a grade of importance to the relations based on the distance between words. So each field of application could gain benefits from a different weight function. Some examples of weight functions are showed in Fig. 2.

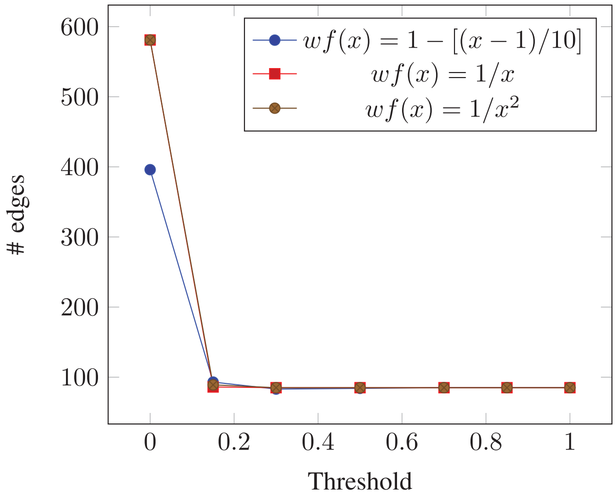

Fig. 3 In this graph three different weight functions are compared based on the resulting edges given a fixed threshold. Each point is an experiment on the verses of the bible as described in Section V. The X axis is the threshold and the Y axis is the number of edges of the resulting graph. All the experiments used a window of size 31

C. Threshold

An edge will be added to the graph only if the score for the relation is bigger than the threshold. This means that the number of edges in the resulting graph depends on the value of the threshold. Since the score is always positive and its maximum value for a document is known, then the threshold can be expressed as a value between 0 and 1. This value will be multiplied for the max score computed as in Section IV-D.

This information can be useful to understand the coherence of the weight function with respect to the corpus. Indeed, if the number of edges doesn't change when the threshold changes, then the weight function could have values too low in the tail of the window. Comparing them as in Fig. 3 could give also information about the corpus. In the figure is clear that almost all the edges are from low score edges, so there are no strong relations in this corpus.

D. Max Score

The maximum cooccorrence score for two words, is given by a document composed only by the two words always one after the other. For instance, two words: A and B, would have the maximum score in a document as the following:

A B A B A B A B A B A B A B ...

In this case, the maximum score is equal to

$$max\_score = \sum_i (|D| - i) \cdot W_i \tag{1}$$

where D is the document as vector of words and W is the vector of weights.

E. Graphs Merging

Since a corpus is composed by many documents, the result of this method would be a series of graphs. To have a single graph for the whole corpus, the graphs should be merged in one. The resulting graph should contain the union of the nodes sets and the union of the edges sets for all the graphs. Except that an edge could be in many graphs, each with a different value.

Even if it is the same edge, the information about which graph contains the edge could be important to perform other analysis. But keep the distinction about only some edges ignoring the others will bring to have a partial information. So the safest procedure is to keep both the original graphs and the global one.

The merging of the graphs is performed just adding the weights for the edges in more than one graph as in algorithm 2.

Algorithm 2 Graph merging

```
function GRAPHMERGE(graphs)
    merged ← new Graph()
    for all $g \in graphs$ do
        for all $e \in g.edges$ do
            $l \leftarrow e.in$
            $r \leftarrow e.out$
            if $e \notin merged$ then
                $merged[l, r] \leftarrow 0$
            end if
            $merged[l, r] \leftarrow merged[l, r].weight + e.weight$
        end for
    end forreturn merged
end function
```

F. Digraph

An important information about words relation is the order of them in the document. There is a big difference between *"The boy also eats the apple"* and *"The apple also eats the boy"*, but without considering the order the resulting graph would be the same (except that the edges with *also*). To maintain the information about the apples, usually to be eaten instead to eat, the cooccurrence graph should be a digraph.

In this case, the edge exits from the left word and enters to the right word. This means that, in the general case, there would be two edges between two words. If only one edge is present between the words *A* and *B*, then these two words are always in the same order in the corpus.

The only difference with respect to the previous algorithm is the sorting of the words. Indeed, the edge is just saved as the order it is encountered in the text.
This type of relation can be used to infer many types of information. For instance, if there exists an edge with a high score between two words, but it doesn't exist the inverse edge, then the two words could be a composite name (for instance *Coca Cola*) or maybe part of a figure of speech (*The white house* means the USA government).

V. An Example

We applied this method on the verses of the bible. We used as nodes the major characters of the bible [24] ignoring all the other words in the documents. Starting from a JSON format [25] we create both versions of the graphs:

- weighted graph based on binary presence of the words
- weighted graph based on the distance of the words

Algorithm 3 Weighted cooccurrence DiGraph

```
function WEICOOCDIGRAPH(words, weights, threshold)
    graph ← new DiGraph()
    scores ← new Dictionary()
    for $i = 0;\ i < words.size();\ i++;$ do
        $left \leftarrow words[i]$
        $maxj \leftarrow \min(i + weights.size() + 1,\ words.size())$
        for $j = i + 1;\ j < maxj;\ j++;$ do
            $right \leftarrow words[j]$
            $l \leftarrow left$
            $r \leftarrow right$
            $k \leftarrow j - i - 1$
            $weight \leftarrow weights[k]$
            $scores[l,\ r] \leftarrow scores[l,\ r] + weight$
            if $l \neq r$ & $scores[l,\ r] > threshold$ then
                $graph[l,\ r] \leftarrow scores[l,\ r]$
            end if
        end for
    end for
    return graph
end function
```

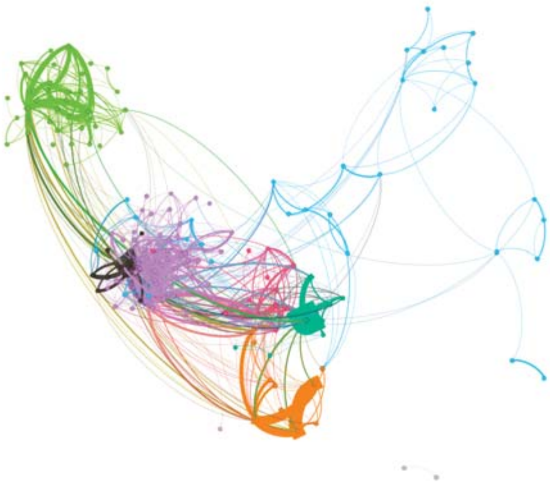

Fig. 4 Cooccurrence graph based on counting verses containing the entities of the bible. See detail in Fig. 5

All the graphs' figures was made using *Gephi* [26] with *OpenOrd* clustering. The overall pictures uses clustering with 750 iterations, while detail pictures uses clustering with 2000 iterations.

A. Standard Binary

In this type of graph, the weight of the edges is equal to the number of documents containing both words as defined in [7]. The result is already a merged graph for all the verses in the bible, so no other merging step was performed. The resulting graph is composed by 180 nodes and 897 edges.

The overall structure is showed in Fig. 4. In this representation we can see 2 big clusters (showed in detail in Fig. 5), 7 little clusters and many other isolated nodes.

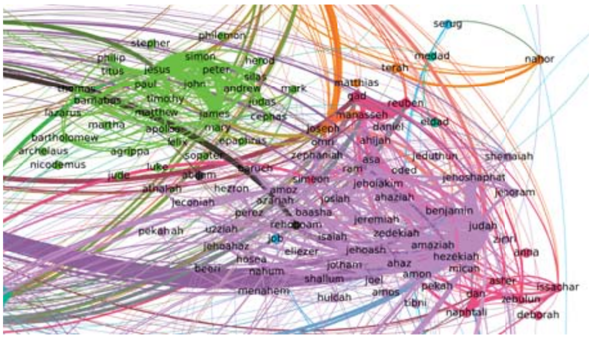

Fig. 5 Two clusters of the cooccurrence graph based on counting verses containing the entities of the bible

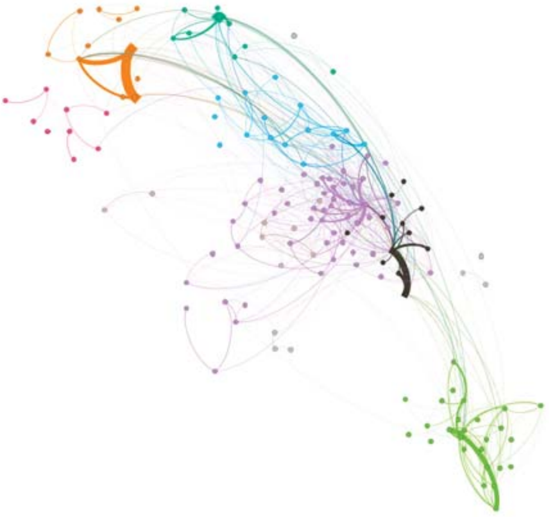

Fig. 6 Cooccurrence graph of entities in the bible, based on a sliding window of size 31 with weight function $1/x$ and threshold 0.0005. See detail in Fig. 7

B. Weighted Window

In this version of the graph we have iterated the algorithm 1 on each verse of the corpus, creating a cooccurrence graph for each verse. We used a window of 31 words with weights computed as $w(d) = 1/d$ (where d indicates the distance from the left word) and a threshold of 0.0005% of the max score. Then we merged the graphs using the algorithm 2. The resulting graph is composed by 162 nodes and 580 edges.

The overall structure is showed in Fig. 6. In this representation we can see 2 big clusters (showed in detail in Fig. 7), 7 little clusters and many other isolated nodes.

C. Comparison of the Graphs

Comparing the graphs we can see many properties that differ in a sensible manner as shown in Table I. In Table II we can see the dimension of the graphs generated by both methods depending on the division of the text. The number of edges are lesser for the weighted window graphs for each division, keeping the weighted window graphs sparser then the binary graphs.

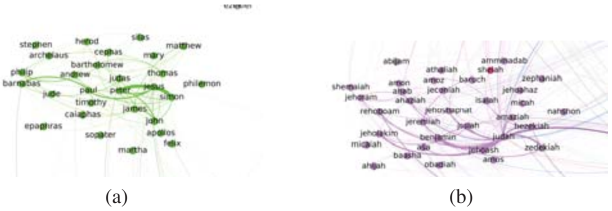

Fig. 7 Two clusters of the cooccurrence graph of entities in the bible, based on a sliding window of size 31 with weight function $1/x$ and threshold 0.0005. (a) This cluster represents characters in *New Testament*, where the biggest edges are: *Jesus* - *Peter* and *Jesus* - *Simon*. (b) This cluster represents characters in *Old Testament*, where *Judah* acts as a "central node"

TABLE I
PROPERTIES OF COOCCURRENCE GRAPHS

Property	Binary	Window
Average degree	9.967	7.16
Network diameter	7	8
Graph density	0.056	0.044
Modularity	0.582	0.666
Connected components	2	5
Average clustering coefficient	0.566	0.536

Comparing, instead the degree of the nodes (as shown in Figs. 8 and 9) the weighted window graph seems to keep the basic structure of the standard binary version. Analysing both the properties and the degrees graphs, the weighted window graph seems to be a more compact version of the standard binary even if the basic structure is maintained.

VI. CONCLUSION

In this paper we present a technique to extract relational information among entities from textual unstructured data (possibly textual documents, or text streams). Following previous works in this area, the idea is to extract word cooccurrence statistics from text, building up a cooccurrence graph. The new idea here is to "weight" distances between words within a sliding window running over the text to be analysed.

Based on the preliminary experiments reported in this paper, the resulting cooccurrence graph contains nearly the same information (number of cooccurrence arcs) of using a simple sliding window, which - in our approach - is the special case where the distance function is a flat constant within the sliding window. But, generalizing over the unweighted case, in our case we have the flexibility of tuning a weight function which is better suited to the nature of the unstructured data, without any loss in performances with respect a "flat" sliding window. The running time is essentially linear in the size of scanned text (the input), and a space which is linear in the size of the total number of retrieved cooccurrences (the cooccurrency graph - i.e., the output - in case of a threshold 0). Hence, this algorithm has performances well suited to scan optimally high

TABLE II
SIZE (NODES AND EDGES) OF COOCCURENCE GRAPHS

Division	Binary		Window	
	Nodes	Edges	Nodes	Edges
Chapters	185	4206	127	172
Verses	180	897	162	580

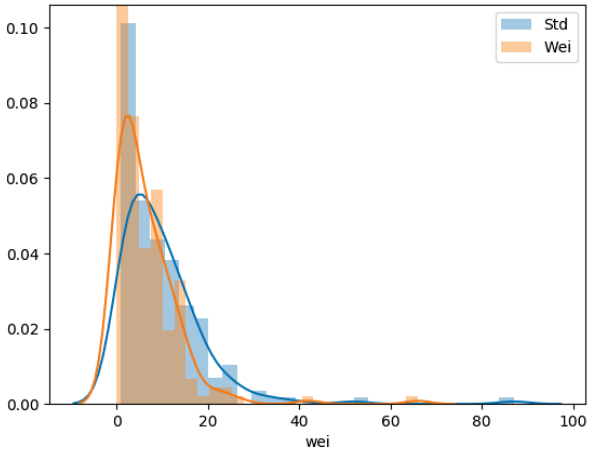

Fig. 8 Distribution of the degree of the nodes for the two cooccurrence graphs computed in Section V. On the X axis there is the degree of a node. Both distributions: standard binary (*std*) and weighted window (*wei*) are plotted

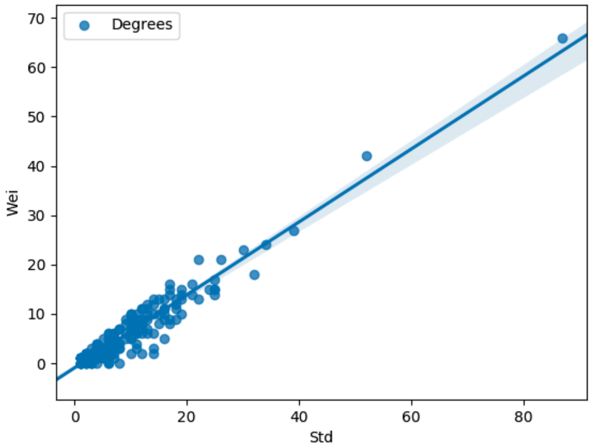

Fig. 9 Scatterplot of the degrees of the entities in the cooccurrence graphs computed in Section V. Each point is a node, in the X axis it is represented its degree in the standard binary graph, while in the Y axis it is represented its degree in the weighted window graph

volume of unstructured data, even from massive text streams, scanned just once, with a memory footprint depending on the "vocabulary", and no dependence on the size of the scanned text.

Based on the performance figures and the flexibility of the proposed data structure, in further work we propose to explore the possible applications of this approach, based on a weighted-distance sliding window, to several contexts in information retrieval and knowledge management, such as: search engine empowerment, thesaurus extraction and verification, ontology maintenance, sentiment analysis, information extraction and document classification.

References

[1] "10 key marketing trends for 2017," https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=WRL12345USEN, accessed: 2018-05-15.

[2] E. Agichtein and L. Gravano, "Snowball: Extracting Relations from Large Plain-Text Collections," *Proceedings of the fifth ACM conference on Digital libraries - DL '00*, vol. I, no. 58, pp. 85–94, 2000.

[3] E. Agichtein, L. Gravano, J. Pavel, V. Sokolova, A. Voskoboynik, E. Agichtein, L. Gravano, J. Pavel, V. Sokolova, and A. Voskoboynik, "Snowball: a prototype system for extracting relations from large text collections," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data - SIGMOD '01*, vol. 30, no. 2. New York, New York, USA: ACM Press, 2001, p. 612.

[4] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.-R. Wen, "StatSnowball : a Statistical Approach to Extracting Entity," *Proceedings of the 18th international conference on World wide web (WWW2009)*, pp. 101–110, 2009. (Online). Available: http://www2009.eprints.org/11/1/p101.pdf.

[5] H. Cunningham, "Information Extraction, Automatic Introduction: Extraction and Retrieval," *Oxford: Elsevier. Heath S B Kortmann B Miller J*, vol. 5, pp. 665–677, 2006. (Online). Available: http://www.elsevier.com/locate/permissionusematerial.

[6] P. Pantel and M. Pennacchiotti, "Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations," *Acl2006*, no. July, pp. 113–120, 2006.

[7] A. Özgür, B. Cetin, and H. Bingl, "Co-Occurrence Network Of Reuters News," *International Journal of Modern Physics C*, vol. 19, no. 05, pp. 689–702, may 2008. (Online). Available: http://www.worldscientific.com/doi/abs/10.1142/S0129183108012431.

[8] H. Sayyadi, M. Hurst, and A. Maykov, "Event Detection and Tracking in Social Streams *," *Proceedings of the Third International ICWSM Conference (2009) Event*, 2009. (Online). Available: https://www.aaai.org/ocs/index.php/ICWSM/09/paper/viewFile/170/493.

[9] K. Tanaka-Ishii and H. Iwasaki, "Clustering Co-Occurrence Graph based on Transitivity," *Fifth Workshop on Very Large Corpora*, 1997. (Online). Available: http://www.aclweb.org/anthology/W97-0111.

[10] D. Benz, C. Körner, A. Hotho, G. Stumme, and M. Strohmaier, "One Tag to Bind Them All: Measuring Term Abstractness in Social Metadata," in *Extended Semantic Web Conference*. Springer, Berlin, Heidelberg, 2011, pp. 360–374. (Online). Available: http://link.springer.com/10.1007/978-3-642-21064-8{_}25.

[11] J. Véronis, "HyperLex: lexical cartography for information retrieval," *Computer Speech & Language*, vol. 18, no. 3, pp. 223–252, jul 2004. (Online). Available: https://www.sciencedirect.com/science/article/pii/S0885230804000142.

[12] E. Agirre, D. Martínez, O. López De Lacalle, and A. Soroa, "Two graph-based algorithms for state-of-the-art WSD," *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pp. 585–593, 2006.

[13] R. Mihalcea and P. Tarau, "TextRank: Bringing Order into Texts," *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004. (Online). Available: http://www.aclweb.org/anthology/W04-3252.

[14] Y. Matsuo and M. Ishizuka, "Keyword Extraction From A Single Document Using Word Co-occurrence Statistical Information," *International Journal on Artificial Intelligence Tools*, vol. 13, no. 01, pp. 157–169, mar 2004. (Online). Available: http://www.worldscientific.com/doi/abs/10.1142/S0218213004001466.

[15] Y. Liu, F. Wang, C. Kang, Y. Gao, and Y. Lu, "Analyzing relatedness by toponym co-occurrences on web pages," *Transactions in GIS*, 2014.

[16] X. Zhong, J. Liu, Y. Gao, and L. Wu, "Analysis of co-occurrence toponyms in web pages based on complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 466, pp. 462–475, jan 2017. (Online). Available: https://www.sciencedirect.com/science/article/pii/S0378437116306409.

[17] Z. Zhang and V. Saligrama, "PRISM: Person Re-Identification via Structured Matching," *IEEE Transaction On Pattern Analysis And Machine Intelligence*, 2015.

[18] I. Ali and A. Melton, "Semantic-Based Text Document Clustering Using Cognitive Semantic Learning and Graph Theory," in *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*. IEEE, jan 2018, pp. 243–247. (Online). Available: https://ieeexplore.ieee.org/document/8334465/.

[19] B. Lemaire and G. Denhière, "Incremental Construction of an Associative Network from a Corpus," *cogprints.org*, 2004. (Online). Available: http://cogprints.org/3779/.

[20] K. Ghoorchian, S. Girdzijauskas, and F. Rahimian, "DeGPar: Large Scale Topic Detection Using Node-Cut Partitioning on Dense Weighted Graphs," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, jun 2017, pp. 775–785. (Online). Available: http://ieeexplore.ieee.org/document/7980020/.

[21] L. Jiang, P. Luo, J. Wang, Y. Xiong, B. Lin, M. Wang, and N. An, "GRIAS: An entity-relation graph based framework for discovering entity aliases," *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 310–319, 2013.

[22] D. V. Kalashnikov and S. Mehrotra, “Domain-independent data cleaning via analysis of entity-relationship graph,” *ACM Transactions on Database Systems*, vol. 31, no. 2, pp. 716–767, 2006.

[23] M. Laclavík, Š. Dlugolinský, and M. Ciglan, “Discovering Relations by Entity Search in Lightweight Semantic Text Graphs,” *Computing And Informatics*, vol. 33, no. 4, pp. 877–906, 2015. (Online). Available: http://www.cai.sk/ojs/index.php/cai/article/viewArticle/2242.

[24] “Bible: Characters in the bible,” *Collins Dictionary*. (Online). Available: https://www.collinsdictionary.com/word-lists/bible-characters-in-the-bible.

[25] T. Bodruk, “Bible: Json + xml,” https://github.com/thiagobodruk/bible.

[26] M. Bastian, S. Heymann, and M. Jacomy, “Gephi: An open source software for exploring and manipulating networks,” 2009. (Online). Available: http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154.

Paolo Fantozzi is a researcher at Centre for Transport and Logistics (CTL) of Sapienza University of Rome. In 2016, he received his Master Degree in Engineering in Computer Science at the School of Engineering in Computer Science of Sapienza University of Rome.

Since 2018 is tutor in Italian Olympiads in Informatics (OII).

Luigi Laura qualified in the National Academic Qualification as Associate Professor, in Computer Science and Computer Science Engineering.

Since 2000 he lectured, in Sapienza and Tor Vergata universities, several courses including “Computer Programming”, “Algorithms and Data Structures”, “Computational Complexity and Models”, and “Web-based Systems Design”; he has been the Thesis Advisor of more than one hundred students.

Since 2006 he is the trainer of the Italian team for the International Olympiads in Informatics (IOI). Since 2011 he is also the scientific supervisor of the Italian Olympiads in Informatics (OII). Since 2016 he is the sceintific-technical manager of OII.

Luigi Laura co-authored, with Giorgio Ausiello, Fabrizio dAmore, and Giorgio Gambosi, the book Linguaggi, Modelli, Complessit (nuova edizione), Franco Angeli Editore (2014).

His research interests involve algorithms for massive scale data, with a particular focus on graph algorithms, on-line algorithms and algorithms in alternative computational models, including external memory, streaming and MapReduce. He authored more than sixty international publications, including international journals and refereed conference proceedings. An updated list of his publication can be found in its Google scholar profile.

Umberto Nanni is full professor of Enterprise Information System in “Sapienza” University, President of the Information Engineering bachelor degree in Latina location of “Sapienza”, Director of the Research Center for Distance Education and Technology Enhanced Learning in “Unitelma Sapienza”. He was co-founder and (up to 2015) member of the directorial board of the “Research Centre in Transport and Logistics”.

His research interests and teaching activities include: graph algorithms and their applications (eg., in transportation science, translational research, architecture), elearning models and technologies, data-intelligence.

In 2011-2014 he was Principal Investigator of the European project “eLF - eLearning Fitness”, funded by EACEA within the LLP Programme; in October 2016 he was PC co-chair of the 15th International Conference on Web-based Learning - ICWL 2016.