Ph.D. Thesis

Algorithms for Knowledge and Information Extraction in Text with Wikipedia

Author
Marco Ponza

Supervisor
Prof. Paolo Ferragina

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Department of Computer Science
University of Pisa

October 2018

Reviewer
Dr. Johannes Hoffart
Max Planck Institute for Informatics
Saarbrücken, Germany

Reviewer
Dr. Dafna Sheinwald
IBM Research
Haifa, Israel

To the perseverance and determination
of my lovely sister, Giorgia.

Abstract

This thesis focuses on the design of algorithms for the extraction of knowledge (in terms of entities belonging to a knowledge graph) and information (in terms of open facts) from text through the use of Wikipedia as main repository of world knowledge.

The first part of the dissertation focuses on research problems that specifically lie in the domain of knowledge and information extraction. In this context, we contribute to the scientific literature with the following three achievements: first, we study the problem of computing the relatedness between Wikipedia entities, through the introduction of a new dataset of human judgements complemented by a study of all entity relatedness measures proposed in recent literature, as well as with the proposal of a new computationally lightweight two-stage framework for relatedness computation; second, we study the problem of entity salience through the design and implementation of a new system that aims at identifying the salient Wikipedia entities occurring in an input text and that improves the state-of-the-art over different datasets; third, we introduce a new research problem called fact salience, which addresses the task of detecting salient open facts extracted from an input text, and we propose, design and implement the first system that efficaciously solves it.

In the second part of the dissertation we study an application of knowledge extraction tools in the domain of expert finding. We propose a new system which hinges upon a novel profiling technique that models people (i.e., experts) through a small and labeled graph drawn from Wikipedia. This new profiling technique is then used for designing a novel suite of ranking algorithms for matching the user query and whose effectiveness is shown by improving state-of-the-art solutions.

Contents

Acknowledgments

I thank my supervisor Paolo Ferragina for his great guidance throughout my Ph.D. studies. He was always ready to offer ideas, advices as well as his huge research experience for the accomplishment of the works described in this dissertation. This thesis would have not been possible without his encouragement and strong belief of the research we did together in these years. I am also honoured to have contributed to the scientific literature in a work together with Soumen Chakrabarti. His insights, suggestions and comments were something extremely invaluable that clearly brought our research to a next level. I also thank the two reviewers of my thesis, Johannes Hoffart and Dafna Sheinwald, for the insightful comments that allowed me to greatly improve this dissertation in its final version.

I am particularly grateful to Gerhard Weikum for having let me join his research group for six months. During that period, I met a number of impressive people, such as the (ex) Ambiverse team, Luciano, Johannes and Dragan, and the D5 members. Particular thanks go to Arunav (the greatest hub of networking ever met), Dat, Sreyasi, Simon R., Paramita, Asia, Azin, Oana, Cuong, Preethi, Kashyap, Jyotsna, Andrew and Daria for the unforgettable dinners, table tennis matches and fun that we had outside of our working hours. Special thanks go to Mojtaba, for all the weekend-trips in the Saarland, and to Krzysztof, for sharing his reckless and adventurous spirit with us.

During my Ph.D. I have the opportunity to meet people coming from all around the world. Particular thanks go to Matthew, Diana, Ioannis, Ekaterina, Jaspreet, Simon G., George, Yagmur, Aldo, Sumit, Anastasia, Zeno, Anna, Emilia, Laura K., Soroush, Ida, Dominic, Gaurav, Ignacio, Manuel and Philip for the few, but extraordinary and intense days that we spent together during the conferences.

The whole Ph.D. would have been much less fun without the support of my two "academic brothers", Francesco and Marco, and the afternoon breaks of the "Paccari" group composed by Giovanna, the two Andrea, Tiziano, Lillo, Daniele, Maria Teresa, Alessio, Lucia, Ottavio, Federico, Laura G., Rita, Nicola and Giacomo. Special thanks go to Veronica and Paolo C., for their patience in the proof-of-reading of this dissertation in its first version.

Most importantly, I warmly thank parents, Antonio and Marilena, and my sister, Giorgia, for their support and unconditional love during these frustrating, challenging, joyous and satisfactory Ph.D. years.

0 Introduction

Academic and industrial research are now focusing on enhancing the humankind progress through the development of new ICT technologies that are considered "intelligent" because they can afford a number of general- or specific-domain tasks in a very accurate manner with performance that is close, and sometimes even better, than what a human can do.

For the development of these intelligent technologies, machines need to access, read and understand the large amount of information stored in data archives. Natural language is still the dominant form on which information is produced every day by humans. News articles, Web pages, emails and social media posts are a few examples of this phenomena.

Text understanding is very easy for humans, but for machines it is still a challenging task. There are many reasons for this: (1) texts do not present a uniform structure; (2) a single sentence can contain multiple facts — e.g., *"Leonardo is the scientist who painted Mona Lisa"* contains the facts *("Leonardo", "is", "scientist")* and *("Leonardo", "painted", "Mona Lisa")*; (3) words are ambiguous since they can refer to multiple real-world entities (e.g., the standalone word *"Leonardo"* could refer to either `Leonardo_DiCaprio` or `Leonardo_da_Vinci`), which become conceptually unique when dived within a specific context (e.g., in the previous example the word *Leonardo* uniquely refers to `Leonardo_da_Vinci`. Furthermore, not all the information present in a document has the same importance: some elements may be more or less salient with respect to the document content, with the top-salient ones that significantly help humans in understanding the discussed topics.

The main reason why humans can easily understand the meaning of a text is that they are able to interpret words and entities in a larger context hinging onto their background and linguistic knowledge (Gabrilovich & Markovitch, 2007). Differently from machines, they do not interpret the meaning of a text by its merely words, but through the salient facts and the concepts they refer to as well as the relatedness among them. In fact, not only are humans able to understand the real-world entities behind an input text, but they are also able to proper quantify how many entities, or facts, are semantically related to another one (e.g., in the previous example, `Mona_Lisa` is more related to `Leonardo_da_Vinci` than to `Scientist`). Unfortunately, most of the solutions present in literature are still based on traditional techniques, whose algorithms are designed on a text representation built upon the classic bag-of-words paradigm (BoW, in short): an unstructured collection of (possibly ambiguous) keywords. Despite the widespread popularity of BoW since sixties (Harris, 1954), this paradigm actually suffers from several well-known limitations: (1) the curse of dimensionality, a problem that arises in algorithms when they need to manage large, and often sparse, data in high-dimensional space; (2) words are ambiguous and afflicted by synonymy and polysemy problems, thus making (3) the understanding of the unambiguous concept they refer to a difficult task; (4) due to the flattering of the input text into a vector of keywords, the set of factual information contained in the sentences is totally lost.

A first effort to overcome these limitations was introduced by algorithms based on matrix-decomposition methods, namely LDA/LSI (Hoffman et al., 2010), with the aim of reducing the dimensions of the input space and building a low-rank approximation of the keyword-document matrix. The resulting space on which documents are transformed is called *latent*, since it manipulates *hidden* concepts that try to model the ones present in the input text and mapped by humans. Conceptually similar words are mapped into similar (geometrically close) latent vectors, thus solving the limitations (1) and (2) mentioned above, but still unsolving (3) and (4).

Due to the scalability issues of LDA/LSI methods and with the progress of the machine learning, research is now focusing on more sophisticated and efficient techniques for learning distributed and dense representations of documents, such as *word embeddings* (Mikolov et al., 2013b). The larger[*] and denser space on which words are now manipulated enables machines to a more semantic (e.g., word analogy) representation of text, thus now addressing (2) and (3), but yet unsolving (1) and (4).

On the other hand, both LDA/LSI and word embeddings only *partially* solve issue (3): words referring to the *same* meaning (e.g., *"NYT"* and *"'New York Times'*) are mapped into *similar* vectors although they actually represent an *equivalent* concept (i.e., the newspaper `The_New_York_Times`).

Consequently, moving toward more structured paradigms is becoming an emerging need for designing new and more intelligent applications that require a deep understanding of the input text, both in terms of unambiguous *salient entities* and *salient facts* expressed in its content. Accordingly, to understand natural language as humans do, machines should be able to access and use a vast amount of common-sense and domain-specific *world knowledge* (Hassan & Mihalcea, 2011). In Computer Science, such as repository is referred as knowledge graph (KG): a graph whose nodes are entities, namely "concepts" representing real-world persons, locations or things, and edges that model their relationships, by properly representing attributes, events or facts between the KG's nodes. Among the number of public available KGs, such as YAGO (Hoffart et al., 2013), BabelNet (Navigli & Ponzetto, 2012) and Wikidata (Vrandečić & Krötzsch, 2014), just to mention a few, Wikipedia is the leading representative and most used repository storing the machines' world knowledge, with a myriad of downstream applications in different domains, such as the ones studied in this dissertation.

Beside accessing to a knowledge repository, machines should be also able to use *linguistic knowledge* of the language at the hand to proper extract from an input text the multiple facts it conveys. A sentence is not only understood through its concepts, but also through the set of propositions (i.e., subject-relation-object) that constitute the input text. Accordingly, the syntactic analysis provided by linguistic frameworks, such as part-of-speech taggers (Manning, 2011), named entity recognizers (Seyler et al., 2017) and dependency parsers (Chen & Manning, 2014), can enable machines to automatically identify the grammatical structure of a text. This linguistic knowledge can thus be used to determine which facts compose a sentence, and consequently help machines to distill, from possibly long and noisy sentences, an

[*]Each word is represented by a n-dimensional vector of floating-points, with n that commonly spans in $[100, 1000]$.

essential and comprehensive set of propositions in normalized form that are extremely easy to be read and understood.

In the last decade, the development of knowledge graphs and advancements in the field of natural language processing allowed research to do several steps towards enabling machines to understand the meaning of input text as well as to structure it in factual form, thus ease its automatic reading and interpretation by properly addressing issues (3) and (4) described above. In particular, two novel research fields have been devised in the domains of, respectively, knowledge and information extraction: *entity linking* (Bunescu & Paşca, 2006) and *open information extraction* (Banko et al., 2007).

Entity linking addresses the problem of detecting meaningful sequences of words in a natural language text and then link them to the nodes of a KG, thus allowing machines to find the concepts (entities of the KG) that can be associated to the input text. Since the text is now mapped into nodes of a KG, machines can perform inference or possibly find the relations between texts based on these entities.

Open information extraction addresses the problem of extracting, from natural language text, of a set of facts in a propositional form (i.e., subject-relation-object). This avoid machines to directly work with sentences, which might express more than one fact, it helps them to comprehensively understand what are the relationships between the extracted constituents as well as it can serve as input to a number of downstream applications — e.g., see (Mausam, 2016) and references therein.

Although the literature abounds nowadays of many theoretical studies about the topics described above and their applications to a variegate number of domains, the same was not true at the time the research pursued in this thesis was started. First, very few works were applying *graph theory* to the field of entity linking with the final goal of *going beyond traditional paradigms* (i.e., BoW, LDA/LSI or word embeddings) that represented a document by means of a vector built upon its words. These few papers proposed new kinds of graph representations that aimed at modeling an input document as a *subgraph* of the KG enriched with proper features (Ni et al., 2016; Scaiella et al., 2012). At the same time, literature was also offering a large body of work addressing the problem of estimating the relatedness between *words*, but the research studying relatedness between *concepts*, which is at the core of any *inference engine* working on KGs, was very preliminary and mainly evaluated in extrinsic settings (Ceccarelli et al., 2013), with solutions hinging only upon the textual description of entities (Hoffart et al., 2012). As a result, machines were not exploiting the full potential provided by both the textual descriptions and the interconnected structure of KGs. On the other hand, several researchers preliminary started using facts as an intermediate representation between the input text and the downstream application at the hand (Stanovsky et al., 2015), thus showing that the extracted facts can help machines in a better comprehension of the document content. Unfortunately, in all these applications, facts have always been considered equally important, whereas it is obvious that some of them are actually more or less salient with respect to the topics discussed in the input text and that this could be deployed within NLP/IR tools in order to empower the understanding of texts.

In this dissertation we contribute to the scientific literature by investigating three technical problems related to *entity linking*, *entity salience* and *fact salience*, which constitute the backbone of many modern NLP/IR tools and naturally spur from the discussion above. Then we introduce and analyze an application setting, called *expert finding*, where we investigate the deployment of our newly designed techniques.

First, we study the problem of *entity relatedness* by introducing a new dataset with human judgements complemented by a thorough study of all entity relatedness measures proposed in recent literature. This allows us to fix the state-of-the-art in this setting, by highlighting pros and cons of the known solutions and, hence, 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 by random walks executed on this subgraph. Our framework shows better agreement with human judgment than existing proposals both on the new dataset and on an established one, with improvements of, respectively, +7% and +5% among Spearman and Pearson correlations. We also plug our relatedness algorithm into a state-of-the-art entity linker and observe an increase in its accuracy (with an average +2/3% in F1 over four different datasets) and robustness (these improvements are stable among different values of a critical system parameter). As a result, our work shows that the combination of textual descriptions and relationships between entities can achieve significantly better results with respect to the methods that rely their computations on only one of these two source of information. This work was presented at *CIKM 2017* (Ponza et al., 2017a) .

Second, we study the problem of *entity salience* through the design and implementation of Swat, a system that identifies the salient Wikipedia entities occurring in an input document. Swat consists of several modules that are able to detect and classify on-the-fly Wikipedia entities as salient or not, based on syntactic, semantic and latent features properly extracted via a supervised process which has been trained over millions of examples drawn from the New York Times corpus. The design of our solution is complemented with a thoughtful analysis that sheds the light among the number of features that have been proposed, by eventually showing that signals coming from a document modeled as a subgraph of the KG can effectively be used to improve the detection of salient Wikipedia entities in texts. The validation process is performed through a large experimental assessment, eventually showing that Swat improves known solutions over all publicly available datasets, with improvements of +3.4% and +6.3% with respect to the systems implemented by Dunietz & Gillick (2014) and Trani et al. (2018), respectively. This work was presented at *NLDB 2017* (Ponza et al., 2017b), and further extended in a version which was published at *Computational Intelligence* (Ponza et al., 2018b).

Third, we introduce *fact salience*, the task of generating a machine-readable representation of the most prominent information in a text document as a set of facts. We also present SalIE, the first fact salience system known in the scientific literature. SalIE is unsupervised and knowledge agnostic, based on open information extraction to detect facts in natural language text, PageRank to determine their relevance, and clustering to promote diversity. SalIE shows that a proper extraction of salient facts can enable machines the understanding of the input document with performance that are near, and sometimes better, than state-of-the-art document summarizers (Durrett et al., 2016; Mihalcea & Tarau, 2004), with improvements up to +2.7%, +3.1%, +1.5% and +1.2% among rouge-1, rouge-l, rouge-1.2w and rouge-su, respectively. This work was presented at *EMNLP 2018* (Ponza et al., 2018a).

Fourth, we are the first (to the best of our knowledge) in showing that the graph representation of documents — preliminary used by Ni et al. (2016) and Scaiella et al. (2012) — can be extended and used for modeling *people* (as opposite to *texts*) as a subgraph of the KG at the hand. More precisely, we investigate the application of entity linking in the domain of *expert finding* with the design and implementation of Wiser, a novel and unsupervised system for the retrieval of experts. Wiser indexes each expert through a new profiling technique which models her expertise with a small, labeled and weighted graph drawn from Wikipedia. At query time, experts are retrieved by Wiser through a combination of a novel suite of scoring algorithms that uses this new profiling representation for boosting the quality of their results. The effectiveness of our solution is established over a large-scale experimental test on a standard dataset for this task. Wiser achieves better performance than all the other competitors, with final improvements over a state-of-the-art solution (Van Gysel et al., 2016b) of +5.4%, +5.7% and +3.9% in MAP, MRR, and NDCG@100, thus proving the effectiveness of author's profile via our "semantic" graph of entities. This work was published at *Information Systems* (Cifariello et al., 2019).

In conclusion, the scientific contributions of this thesis show that the proper modeling of knowledge and information extracted from an input document in terms of entities, drawn from a KG (e.g., Wikipedia), and facts, derived from its sentences, can be significantly boosted through the application of graph modeling and algorithms. Specifically, structuring the knowledge and information contained in a document as a graph (of entities or facts) allows the design of new and powerful techniques and novel NLP/IR tools which target the ultimate and ambitious goal of enabling machines to comprehensively understand and interpret the natural language text.

0.1 Thesis Outline

The thesis is organized in three main parts.

Background and Tools introduces several basic concepts and tools that serve as building blocks for the rest of the thesis. Specifically, it is subdivided in two main chapters.

Chapter 1 illustrates three main research areas. Each of them is first described in general terms and then specialized with a suit of practical applications, thus showing their role in the design of "more intelligent" solutions. Specifically, the three research areas are: (1) Wikipedia as fundamental resource for the developing of intelligent systems based on real-world knowledge; (2) knowledge extraction, with a particular focus on the domain of entity linking; (3) information extraction, with special emphasis on open information extraction. In this chapter — and, specifically, in (2) and (3) — we also detail the differences that feature solutions for the extraction of knowledge with respect to the ones that extract information.

Chapter 2 concludes this first part by introducing four main algorithmic techniques that are widely used in the rest of the thesis. Specifically, we describe (1) the theory of graph algorithms based on random walks (i.e., PageRank and CoSimRank); (2) the popular word embeddings technique, commonly used for learning the latent distribution of words and entities in a text collection; (3) the well-known machine learning algorithm for classification and regression problems known as gradient tree boosting; (4) a novel and efficient clustering algorithm for partitioning data points into high-quality clusters without sacrificing the scalability performance.

Knowledge and Information Extraction illustrates three main technical contributions of this thesis, subdivided in two chapters.

Chapter 3 formalizes the problem of quantifying how much two Wikipedia entities are related through a fair and unbiased study of all relatedness measures proposed in recent literature over two different human assessed datasets (one of which is new and introduced in the same chapter). In this context, we then propose a new, space-efficient, computationally lightweight two-stage framework for the computation of such entity relatedness. Experimental results will show that our framework not only achieves a better agreement with human judgement than existing proposals, but it also allows a state-of-the-art entity linker to increase its accuracy and robustness.

Chapter 4 is focused on the extraction of salient elements (i.e., entities and facts) from an input document. The first part of the chapter lies in the domain of *entity salience*, which addresses the problem of extracting salient Wikipedia entities from text. In this context, we present the design and implementation principles of Swat, our novel state-of-the-art solution for the entity salience problem. The second part of this chapter introduces a new task called *fact salience*, which addresses the problem of extracting salient facts from an input text. After introducing this new research problem, we present SalIE, the first, unsupervised and knowledge agnostic solution for the extraction of salient facts.

Applications is the part of the thesis focused on showing the practical impact that knowledge extraction tools can have in the domain of *expert finding*. More precisely, Chapter 5 focuses on *expert finding*, a task whose goal is the retrieval of pertinent people (i.e., experts) given an input query. We describe the design and implementation of Wiser, our novel expert finding solution. In this context, Wiser shows how entity linking and the relatedness between the extracted entities can be used to properly model the expertise of people through a small, labeled and weighted graph of Wikipedia entities. Furthermore, this new profiling technique is used to design a new suite of ranking algorithms for the retrieval of experts, whose effectiveness is established over a large-scale experimental test that eventually shows Wiser turning out to be state-of-the-art on this task.

0.1.1 Internal Structure of the Chapters

All chapters that introduce our research contributions generally follows this structure:

1. At the very beginning, each chapter provides a brief and general abstract of the work that is going to be presented.
2. Through the section *Introduction*, the problem and the main contributions of the chapter at the hand is illustrated.
3. A section *Related Work* is specifically provided for each chapter.
4. Subsequently, one or more sections detail the novel contributions that are devised in the chapter at the hand.
5. *Experiments* is the section that describes the datasets, configurations of the experimented systems, evaluation metrics, results and discussion spurred out from our experiments.

Since Chapter 4 groups together two different works on the common topic of extraction of salient elements (i.e., entities and facts) in text, it follows this structure at a deeper level than other chapters. More precisely, points 1-3 are preserved, but 4-5 are actually presented within two different sections.

0.2 Research Contributions

We list here the contributions published, or still under review, of the author of this dissertation, properly sorted by their date of appearance.

M. Ponza, F. Piccinno and P. Ferragina. Document Aboutness via Sophisticated Syntactic and Semantic Features. In *Proceedings of the 2017 International Conference on Natural Language and Information Systems. NLDB 2017, pages 441–453, Lecture Notes in Computer Science, Springer.*

M. Ponza, P. Ferragina and S. Chakrabarti. A Two-Stage Framework for Computing Entity Relatedenss in Wikipedia. In *Proceedings of the 2017 International Conference on Conference on Information and Knowledge Management, CIKM 2017, pages 1867–1876, ACM.*

M. Ponza, L. Del Corro and G. Weikum. Facts That Matter. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018, pages 1043–1048, ACL.*

M. Ponza, F. Piccinno and P. Ferragina. Swat: A System for Detecting Salient Wikipedia Entities in Texts. *Computational Intelligence 2019, Wiley-Blackwell Publishing*.

P. Cifariello, P. Ferragina and M. Ponza. Wiser: A Semantic Approach for Expert Finding in Academia based on Entity Linking. *Information Systems 2019, pages 1–16, Elsevier*.

Part I
Background and Tools

1 Background

In this chapter we introduce several concepts that serve as the fundamental background for the understanding of the dissertation. Specifically, we cover three different broad and wide-spread topics. First, we introduce Wikipedia as resource of knowledge intended for being used to develop intelligent applications as well as how we model Wikipedia in the form of directed and labeled graph. This kind of modeling actually constitute the knowledge graph that we use in the rest of the thesis. Second, we introduce the research area of knowledge extraction, with a proper specialization on the domain of entity linking: the task that addresses the extraction of entities belonging to a knowledge graph from an input text. Third, we describe the topic of information extraction, with particular focus on open information extraction: the tasks that addresses the extraction of open facts from a input text. Differences between the topics of knowledge and information extraction are also provided in the corresponding sections. Furthermore, the description of each background topic is supplemented with a proper section that focuses its use within different research areas, thus showing the effectiveness and importance they have in a myriad of different research communities.

1.1 Wikipedia: The World Knowledge Repository

Empowering intelligent applications with the use of *world knowledge* is becoming a powerful asset not only for increasing the quality of a large number of tasks, but also for improving the user experience as well as for enhancing the interpretability of the resulting solutions. A resource that stores the "world knowledge" is called *knowledge graph* (KG), and a variegated number of them have been proposed in literature. Popular public available examples are Wikipedia[*], DBpedia (Lehmann et al., 2015), BabelNet (Navigli & Ponzetto, 2012), Yago (Hoffart et al., 2013), ConceptNet (Speer & Havasi, 2013), Freebase (Bollacker et al., 2008) and Wikidata (Vrandečić & Krötzsch, 2014), which have been complemented by that vertical catalogs proposed as domain-specific KGs (e.g., movie, medical, sport, finance and more). Industrial examples do also exist, such as the Google Knowledge Graph (Singhal, 2012), Facebook's Graph Search[†] or Microsoft Satori.

From the above mentioned KGs, Wikipedia is clearly the leading representative and most used repository of world knowledge (Pasca, 2016), especially in the academia domain. First, Wikipedia is the one which is commonly chosen as starting point for deriving of other KGs, for example by fusing multiple languages of the same Wikipedia snapshot (Navigli & Ponzetto, 2012) with a proper combination (Hoffart et al., 2013) of other existing KGs, such as WordNet (Miller, 1995) or GeoNames (Vatant & Wick, 2012). Second, according to a recent study (West et al., 2012), Wikipedia is mainly edited by people with a high expertise with respect to a domain of interest as well as they "smarter" than common Web users: they read more news, educational websites and reference sites as well as they are deeply immersed in the pop culture. Third, Wikipedia is a dynamic collection of topics whose resources are in a no-ending growing trend. Articles are incrementally updated, new pages are added every day and the quality of the knowledge present in Wikipedia advances month by month (Bairi et al., 2015). Furthermore, the semi-structured form of Wikipedia allows researchers and engineers to easily model its encyclopaedic knowledge in a machine-readable structure. Textual descriptions, relationship between articles, categories and infobox information can be easily extracted and modeled in the most convenient structure driven from the downstream applications at hand.

[*] wikipedia.org
[†] facebook.com/graphsearcher

1.1.1 Wikipedia in Numbers

Wikipedia is a free, multilingual and openly editable encyclopedia owned by the Wikimedia Foundation[*]. The name Wikipedia is a blend by two words: *wiki*, a Hawaiian word which means fast[†], and *encyclopedia*, a resource that gives information on many topics. From its launch on January 2001, when only the single edition in English language was available, it has been consistently expanded from a community that grows every year. Nowadays, Wikipedia counts a total of 48M of articles in 301 different languages, with a monthly average of 16B of page views and 30M of edits[‡].

In this thesis, we use a KG generated from the English version of Wikipedia, since it is the one with the widest monolingual topic coverage (5.5M articles and 7.5B of page views every month) and it can count the largest number of use cases in different domains and applications (see next section), with a variegated set of publicly available tools that can be easily adaptable to work on the topics covered in this dissertation.

1.1.2 Applications of Wikipedia

From its advent, Wikipedia has been successfully used in a large set of research projects (Pasca, 2016), by ranging a different number of applications that have been implemented in a wide set of domains.

Deriving other Knowledge Graphs. Wikipedia is the main resource used as starting point for the birth of other knowledge graphs. Most of the KGs are constructed by first harvesting the data from Wikipedia, and then they are subsequently extended with the information that comes from different sources. Pioneer examples of Wikipedia-derived KGs are:

[*]wikimediafoundation.org

[†]The term *wiki* was first originally used as name for a website from which an user can edit the content, i.e. wiki.c2.com/?WikiWikiWeb.

[‡]All statistics are taken on May 2018 from stats.wikimedia.org/v2.

- *DBpedia* (Lehmann et al., 2015) is a publicly available resource[*] generated through an extraction framework that converts the contents of Wikipedia (e.g., articles' texts, infobox, categories and geo-coordinates) into a multi-domain knowledge graph of RDF triples according to a different set of heuristics.

- *BabelNet* (Navigli & Ponzetto, 2010, 2012) is multilingual and publicly available[†] semantic network automatically developed from the Sapienza University of Rome. It is built from Wikipedia and WordNet, and further extended through the application of machine translation techniques in order to enrich all languages with proper lexical information.

- *Yago* (Suchanek et al., 2008), which name stands for Yet Another Great Ontology, is a KG developed from the Max-Planck Institute for Informatics (Saarbrücken). This repository is automatically constructed by properly harvesting the Wikipedia categories and then extended with WordNet through a different set of rule-based heuristics. Yago has also been extended onto several dimensions, such as time and space (Hoffart et al., 2013). This KG is also publicly available [‡].

- *ConceptNet* (Speer & Havasi, 2013) has been generated from the Open Mind Common Sense project (Singh et al., 2002) and it is now an open-source project [§] of Luminoso Technologies. It aims at being a multilingual KG that contains both general and domain-specific knowledge. ConceptNet's data is harvested from different resources, such as Wikipedia, DBpedia, Frebase and WordNet.

- *Freebase* (Bollacker et al., 2008, 2007) was an open KG developed in the industrial domain by Metaweb Technologies and later acquired by Google. Freebase contains data initially harvested by Wikipedia and then collaboratively edited by its users. The speciality of this KG was a lightweight typing system designed as a collection of conventions (as opposite to the rigid system of ontologies) where conflicts and contradictions might simultaneously exist in order to possibly reflect users' different opinions and un-

[*]dbpedia.org
[†]babelnet.org
[‡]yago-knowledge.org
[§]conceptnet.io

derstanding. This project has been shut down in 2016 in order to offer its content to Wikidata (Pellissier Tanon et al., 2016), but its last dump is still publicly available[*].

- *Wikidata* (Vrandečić, 2012; Vrandečić & Krötzsch, 2014), whose name stands for Wikipedia for Data, was developed in order to ease the managing the factual information of Wikipedia and overcome several inconsistencies that can be found in that KG[†]. Plurality of facts is allowed as done by Freebase, thus allowing conflicted data to coexist. Wikidata is also publicly available[‡].

Beyond deriving KGs, Wikipedia is also used to generate other resources, such as learning language models (Mikolov et al., 2013b; Ni et al., 2016), fusing the knowledge that comes from different resources (Dong et al., 2014) or the construction of a hierarchy of categories (Aouicha et al., 2016; Boldi & Monti, 2016; Hu et al., 2015).

Natural Language Processing. Wikipedia has been widely used to improve the quality of different NLP tasks. For example, Ponzetto & Strube (2006) propose a coreference resolver whose features are based both on Wikipedia categories and word relatedness (Strube & Ponzetto, 2006). Wu & Weld (2010) develop a distant-supervised information extraction system whose training data is generated by matching Wikipedia infoboxes with the corresponding text, Voskarides et al. (2015) explain relationships between entities in KGs through sentences extracted from Wikipedia, Banerjee & Mitra (2016) propose a summarizer for generating Wikipedia summary from an article's content and Chen et al. (2017) present a question-answering system whose results are retrieved by paraphrasing Wikipedia sentences through a recurrent neural network. A large set of applications of the Wikipedia as KG also lie into the wide domain of knowledge extraction, such as relation extraction (Nguyen et al., 2017b; Ru et al., 2018; Sorokin & Gurevych, 2017; Yan et al., 2009) and entity linking (Bunescu & Paşca, 2006; Ferragina & Scaiella, 2012; Ganea et al., 2016; Nguyen et al., 2014; Zwicklbauer et al., 2016).

[*]developers.google.com/freebase

[†]For example, the information about the population of Italy can be found both in English and Italian Wikipedias in different pages, but the numbers are all different.

[‡]wikidata.org

Information Retrieval. Understanding queries that a user submits to a search engine (Blanco et al., 2015; Cornolti et al., 2016; Tan et al., 2017), items recommendation (Bi et al., 2015; Katz et al., 2011), classification (Lim & Datta, 2013; Pasca, 2018; Vitale et al., 2012), clustering (Hu et al., 2009; Scaiella et al., 2012) and expert finding (Demartini, 2007) are just a subset of the examples where Wikipedia is used within the IR community.

Understanding Human Factors. Beside the use of Wikipedia as the stepping stone on which different tasks have been developed, Wikipedia has also been frequently used as primary source for understanding the human behaviour. For example, research has attempted to show the motivations that bring a user to read Wikipedia (Singer et al., 2017), or what are the profile(s) behind a Wikipedia editor (West et al., 2012) as well as what can be future editors' interests (Yazdanian et al., 2018). Gender gaps (Ross et al., 2018) and inequality (Wagner et al., 2015), navigability (Lamprecht et al., 2016), privacy loss (Rizoiu et al., 2016) are such a few of the topics of interest that have seen Wikipedia as resource of knowledge for different studies (Gandica et al., 2016; Kim & Oh, 2016; Merchant et al., 2016).

1.1.3 Structuring Wikipedia

In this section we formalize the terminology that we use in the rest of the thesis when referring to Wikipedia as our KG. More precisely, we model Wikipedia as a directed and labeled graph $KG = (V, E)$. In our context, a node $u \in V$ is an atomic real-world *entity*, such as person, place or thing, which is uniquely identified in V through its Wikipedia identifier. We interchangeably use the term entity, Wikipedia page or node $u \in V$ for referring to a unique element of our KG. We label every node with a function $label : V \rightarrow String$ in order to provide each node $u \in V$ with the textual description of the corresponding Wikipedia article. We call *Wikipedia corpus* the whole textual knowledge available in Wikipedia that can be derived through $\bigcup_{u \in V} label(u)$. We denote a specific entity with the typewriter font (e.g., `Leonardo_Da_Vinci`), which is also hyperlinked to the proper Wikipedia page in the digital version of the thesis.

1.2 Knowledge Extraction

Knowledge extraction is the task that aims at creating structured knowledge from an arbitrary text in order to ease machines in their understanding and inferring. Literature does not offer a formal consensus between knowledge and information extraction (described in Section 1.3), with these two terms that are sometimes used interchangeably and sometimes they are not. For the sake of clarity, in this thesis we decided to distinguish between them and thus providing a clearer separation when discussing about algorithms developed on the top of KG (i.e., knowledge extraction algorithms) and when discussing about algorithms that do not need of accessing to any background knowledge (i.e., information extraction algorithms).

Technically speaking, our knowledge extraction algorithms are designed within one of its particular subfield, i.e. *entity linking*, which involves the linking of the information extracted from an input text into nodes of a KG. Entity linking is clearly a fundamental task for knowledge extraction, since most of its other subfields are commonly developed on the top of entity linking systems, such as relation extraction (Li et al., 2016; Nguyen et al., 2017b), question answering (Abujabal et al., 2018; Yahya et al., 2013) and knowledge graph population (Ji & Grishman, 2011; Nguyen et al., 2017a; Shin et al., 2015)

1.2.1 Entity Linking

Entity linking is the task that addresses the automatic extraction of knowledge present in an input document by properly connecting the information of its content with the entities of a KG. More formally, the goal of an entity linking system is to detect short and meaningful sequences of words (also called *mentions* or *surface forms*) and to disambiguate their meaning by properly associating to each of them the unambiguous concept that is represented by a node of the KG at the hand.

Example. Given the sentence *"Leonardo is the scientist who painted Mona Lisa"*, an entity linker should detects and links the mention *"Leonardo"* with the Wikipedia entity `Leonardo_da_Vinci` and the *"Mona Lisa"* with the painting `Mona_Lisa`. Entity linking systems should be able to distinguish the meaning of *"Leonardo"* among the inventor `Leonardo_da_Vinci`, the actor `Leonardo_di_Caprio` and the fictional character `Leonardo_(TMNT)`.

Entity Linkers

Although most of the entity linking systems designed in the last years differ in the algorithms they implement, a large number of them can be designed as a pipeline of three main stages (a graphical example is provided in Figure 1.1):

1. *Mention Detection (also knew as Spotting).* This step aims at identifying a meaningful sequence of words (mention) that are used as input for the next stage. Most of the approaches rely the detection of mentions on POS taggers or NER systems (Bunescu & Paşca, 2006; Cucerzan, 2007; Hoffart et al., 2011; Moro et al., 2014; Piccinno & Ferragina, 2014; Ratinov et al., 2011), but algorithms for recall-oriented solutions based on word-grams (Ferragina & Scaiella, 2012; Meij et al., 2012), or keywords (Mihalcea & Csomai, 2007) do also exist.

2. *Candidate Generation.* This second stage aims at generating a set of candidate entities for each mention detected in the previous stage. The standard approach for solving this stage is based on the technique originally proposed by (Bunescu & Paşca, 2006) for building a dictionary D of mentions in the form of inverted list. Given a mention m, the dictionary D returns a set of sorted candidate entities $D(m)$ that can appear in the text through the mention m. An entity $e \in D(m)$ is also ranked according to the *prior probability* (also knew as *commonness*) that the mention m refers to e. Entries of the dictionary and the computation of the prior probabilities are commonly obtained through the use of link anchors, redirection and disambiguation pages of Wikipedia (Shen et al., 2015). Other techniques have been also developed, such as heuristic-based methods (Ratinov et al., 2011), boosting the generation of candidates through Web links (Chisholm & Hachey, 2015) or expanding the context of the input text by piggybacking a Web/Wikipedia-based search engine (Cornolti et al., 2016; Tan et al., 2017), but their algorithms always hinge upon a dictionary of mentions built from Wikipedia.

3. *Disambiguation.* The last stage of the entity linking pipeline aims at assigning to each mention the most pertinent entity selected among its candidates, according to the context of its occurrence. The standard solution for solving this stage is to model the dis-

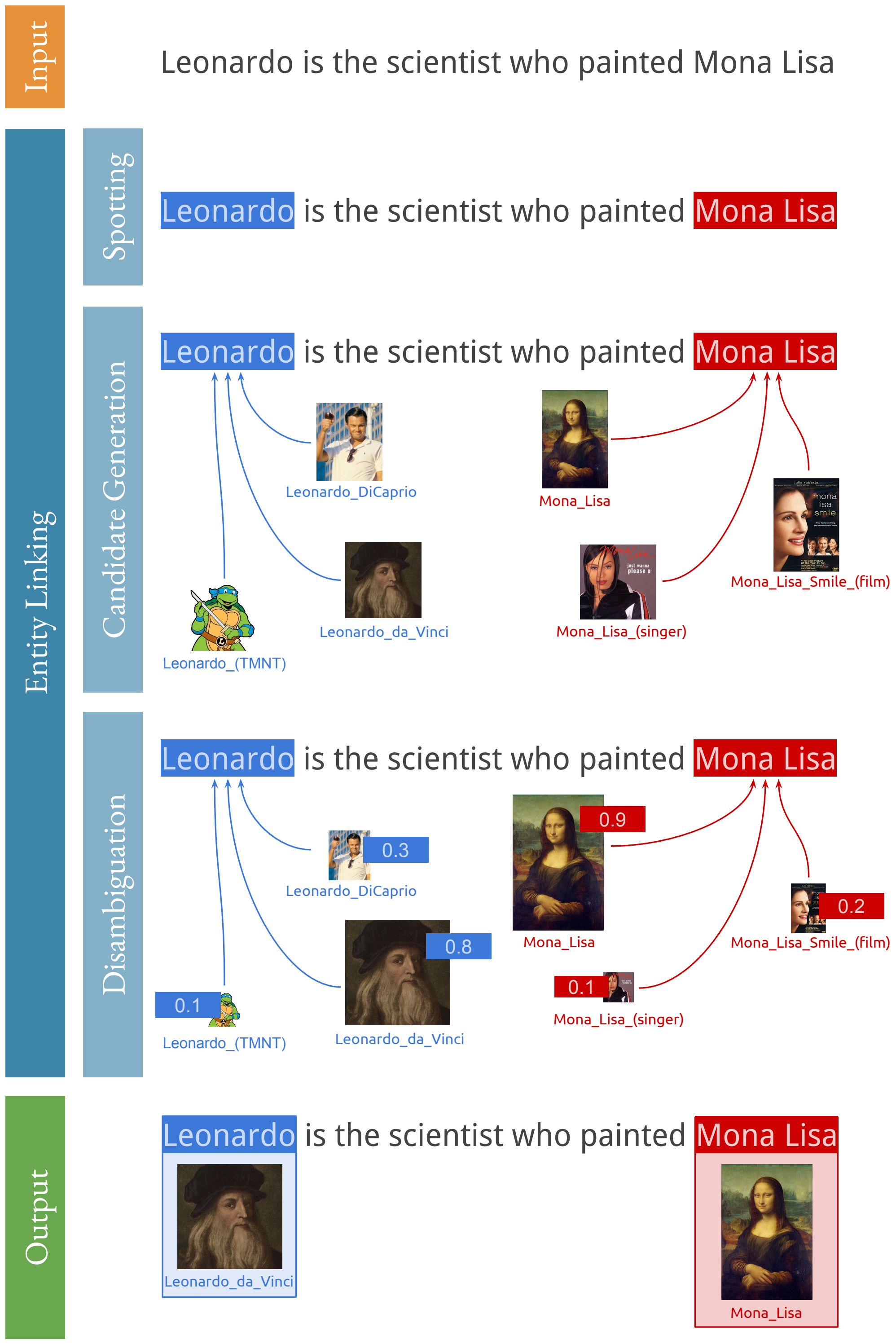


Figure 1.1: Anecdotal example that shows the three main stages implemented by an entity linking system. For lack of space we show a spotting phase that detects only named entities (i.e., *Leonardo* and *Mona Lisa*) as mentions, with a candidate generation phase limited to three possible entities for each of them. Disambiguation phase can be implemented in different ways (see text) and it assigns a pertinence score to each entity. The final output contains only the most pertinent entity associated for each detected mention.

ambiguation problem as a ranking task (Ferragina & Scaiella, 2012; Ganea et al., 2016; Nguyen et al., 2014; Piccinno & Ferragina, 2014; Zwicklbauer et al., 2016) that assigns to each candidate entity e a proper score that represents the pertinence/coherence of e with respect to the input mention and the other entities. Each mention is eventually assigned to the entity with the highest pertinence score among its candidates.

The implementation of the second stage as a dictionary of mentions generated from Wikipedia is a standard practice, whereas for the first and third stage literature abounds of a large number of solutions:

Wikify! (Mihalcea & Csomai, 2007) is one of the first study that addresses the problem of extracting Wikipedia entities from an input text. The mention detection stage is implemented as a keyword extraction algorithm that works at word-gram level in order to extract relevant phrases from the input document. The disambiguation phases involves a naive Bayes classifier that discriminates candidate entities through a feature space designed on the top of local (e.g., term frequency) and topical (e.g., prior probability) entities' attributes (Mihalcea, 2007).

Aida (Hoffart et al., 2011) is an entity linker that deploys Stanford CoreNLP (Manning et al., 2014) as mention detector and a combination of several features for solving the disambiguation stage with a greedy algorithm. More precisely, (Hoffart et al., 2011) models the disambiguation problem as a graph where mentions and candidate entities are nodes and edges are weighted with different similarity measures (Hoffart et al., 2012; Milne & Witten, 2008; Thater et al., 2010). The association between mention-entity pairs is eventually computed by running an extension of an approximation algorithm (Sozio & Gionis, 2010) for the problem of finding strongly interconnected, size-limited groups in social networks. A faster and lightweight version of Aida has been developed by Nguyen et al. (2014), achieving competitive accuracy and a lower running-time. The source code of Aida is publicly available[*].

Rel-RW (Guo & Barbosa, 2014) is an approach based on an iterative mention disambiguation algorithm that works on an expanded graph of candidate entities. The graph construction starts with the set of candidate entities generated from each mention and then it is ex-

[*]mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/aida

panded by adding all entities adjacent to them in the whole Wikipedia graph. The disambiguation algorithm builds a set of semantic signatures (i.e., a set of vectors of stationary distributions for each entity) that are then iteratively updated until each mention is disambiguated. More precisely, at each step, the algorithm disambiguates those mentions which have the candidate entity with the maximum zero-KL-divergence between its semantic signature and the semantic signatures of the already disambiguated mentions.

Babelfy (Moro et al., 2014) is a system that disambiguates entities by combining semantic signatures and subgraph statistics of the candidate entities. The mention detection stage deploys the Stanford POS tagger (Manning et al., 2014) in order to match text fragments with entries present in BabelNet (Navigli & Ponzetto, 2012). Then, a directed and weighted graph is built in which nodes are BabelNet entities and edges are weighted with a score based on the local clustering coefficient (Watts & Strogatz, 1998). Similarly to Hoffart et al. (2011), the disambiguation stage runs a greedy algorithm for detecting the densest subgraph (the subgraph with the highest weighted number of connections among its nodes): the intuition of this algorithm is that the most suitable entity of each mention belongs to the densest area of the graph. Babelfy can be automatically queried via web API[*].

TagMe (Ferragina & Scaiella, 2012) is one of the fastest entity linker to date, able to achieve a competitive performance among several datasets (Usbeck et al., 2015). The mention detection stage identifies the mentions by querying the dictionary D with sequence of words by eventually keeping the longest non-overlapping mentions. The disambiguation stage is solved through a “collective agreement” between entities. Specifically, each candidate entity votes the other entities and the resulting annotations are the top-scored ones. A vote from an entity to another one is computed as the Milne&Witten (Milne & Witten, 2008) relatedness between these two . Because TagMe is the system that offers the best trade-off between accuracy and efficiency (Usbeck et al., 2015), it has continuously been used from a large number of applications (Aprilius et al., 2017; Basile et al., 2016; Chabchoub et al., 2016; Cornolti et al., 2016; Hasibi et al., 2016; Meij et al., 2012; Nanni et al., 2016; Ni et al., 2016; Oramas et al., 2016; Raviv et al., 2016). TagMe is also open-source[†] and its API are publicly available[‡].

[*] babelfy.org
[†] github.com/gammaliu/tagme
[‡] sobigdata.d4science.org/web/tagme/tagme-help

Wat (Piccinno & Ferragina, 2014) is a systems that hinges upon the TagMe's algorithmic technology, but it proposes some improvements over its entity linking pipeline that allow to achieve significantly better performance. More precisely, in the mention detection stage Wat integrates several NLP tools, such as OpenNLP (OpenNLP, 2011) and Stanford CoreNLP (Manning et al., 2014), while the disambiguation algorithm uses the same voting-scheme of TagMe but votes are here calculated through Jaccard relatedness (as opposite to Milne&Witten) and properly combined with PageRank. Wat can be automatically queried via web API[*].

PBoH (Ganea et al., 2016) is a probabilistic approach that makes use of the loopy belief propagation algorithm (Murphy et al., 1999) to perform a collective entity disambiguation via approximate inference. This system exploits co-occurrence statistics in a fully probabilistic manner by using unsupervised machine learning. It relies on statistics counting (hyperlinks frequency and pairwise entity co-occurrence) and it depends on very few parameters. The source code of PBoH is publicly available[†].

DoSeR is an entity linker proposed by Zwicklbauer et al. (2016) works in two main steps by heavily relying its computation on the embedding representation of Wikipedia entities learned via Word2Vec (Le & Mikolov, 2014; Mikolov et al., 2013b). First, candidate entities are filtered by keeping the ones with a high prior probability and which are highly related to the input document. An entity is considered highly related to the input document if the cosine similarity between its embedding vector and the document embedding is greater than a fixed threshold. The second step is the core of the DoSeR's disambiguation algorithm and it works as follows. A directed graph whose nodes are both candidate entities (produced in the previous step) and topical nodes (nodes which represent the average topic of all already disambiguated entities) are generated. Edges are weighted by harmonic mean of the cosine similarity between their embedding and the context where an entity appears. Entities are finally disambiguated by keeping the top-scoring entities after running the PageRank algorithm, possibly in an iterative way. The source code of DoSeR is publicly available[‡].

[*] sobigdata.d4science.org/web/tagme/wat-api
[†] github.com/octavian-ganea/pboh-entity-linking
[‡] github.com/quhfus/DoSeR

For estimating the efficiency and effectiveness of entity linking systems, recently advancements have been achieved, with the proposal of different publicly available datasets and the introductions of the Bat framework[*] (Cornolti et al., 2013), subsequently incorporated into the Gerbil[†] platform (Usbeck et al., 2015), which are now becoming the de-facto standards for the evaluation of the performance of new entity linking solutions.

Applications

Extracting knowledge from textual documents through entity linking is becoming a stepping stone for advancing the quality and interpretability of a large number of software tools in different domains (Bansal et al., 2015; Bhagavatula et al., 2015; Blanco et al., 2015; Dunietz & Gillick, 2014; Ni et al., 2016; Scaiella et al., 2012; Trani et al., 2018; Vitale et al., 2012). Since the extracted knowledge is represented as nodes of a KG, more sophisticated methods can be designed in order to empower classical approaches via new and more intelligent solutions that can benefit of the interconnections and additional meta-data which are available in those KGs (Dietz et al., 2017). Entities' descriptions, categories and the interconnected relationships between entities can now help the downstream application at hand in the understanding of the input text.

Classification. One of the core application in the domain of IR is clearly the classification of texts into a predefined set of categories (Sebastiani, 2002). Several works (Ferragina et al., 2015; Vitale et al., 2012) have preliminary shown the benefits of entity linking in the context of classification of news and hashtags through the designing of new algorithms that rely their computations on different signals (e.g., entities' relatedness and their categories) drawn from the underlying KG.

Clustering. Grouping together documents into topically coherent clusters is another common challenge that often occurs in IR. A preliminary research (Scaiella et al., 2012) has shown how entity linking can be used for the design of a new clustering paradigm that significantly improves in terms of both quality of the clustering and efficiency of the processing the known solutions whenever the texts have short length (as in the case of Web-search snippets).

[*] github.com/marcocor/bat-framework
[†] aksw.org/Projects/GERBIL.html

Relatedness. Measuring how much two input documents are related (semantically similar) is still a challenging and active field for both IR and NLP communities. One of the best proposed systems (Ni et al., 2016) for relatedness computation hinges its technology upon a graph representation of the input documents, properly created through the deployment of entity linking. The relatedness between two documents is here computed by combining several signals that are respectively derived from this graph (e.g., closeness centrality) or from the underlying KG (e.g., entities' description, categories and relationships). Beyond news documents, entity linking has also shown to be effective in the context of short texts for the computation of the relatedness between hashtags (Ferragina et al., 2015).

Query Understanding. Entity linking in queries is quickly emerging as a novel algorithmic challenge (Blanco et al., 2015; Cornolti et al., 2016; Tan et al., 2017) that aims at understanding query intents (partially) expressed by users via few words (i.e., web-search query). The input text is very short and possibly shows misspellings as well as unreliable tokenization, capitalization and word order. Current state-of-the-art systems (Cornolti et al., 2016; Tan et al., 2017) enrich the query's context through a piggybacking technique in order to discover candidate entities and subsequently link-back them to mentions occurring in the query's text.

Summarization. Document summarizers aim at detecting the gist of an input document through computations that are usually based on the information derived from the local content (e.g., frequency) of the input text. Because these computations process the lexical elements (words or phrases) of the document at hand, summarizers commonly incur in several limitations which have been properly highlighted in the previous literature — e.g., see (Hasan & Ng, 2014). *Entity salience* (Dunietz & Gillick, 2014) is a recent introduced task that aims at overcoming these limitations with the extraction of *salient knowledge* present in the input text through the deployment of entity linking. In this context, summaries are no more words or phrases, but a set of unambiguous *salient entities* drawn from a KG. Complementary to classical approaches, entity salience systems leverage their computations on the attributes and relationships between the extracted entities as well as on the signals derived from the local content of the input text. In this thesis, we attack the problem of entity salience with the proposal of Swat (Ponza et al., 2017b, 2018b) , a new system that improves the state-of-the-art over all public available datasets. More details about Swat are given in Chapter 4.

Expert Finding. Most of the downstream applications that we described above have shown how the knowledge extracted with the use of entity linking can be used for improving of the quality of end-to-end results in a variegated set of domains. In the quest of a more accurate expert finding solution, we initially follow this line of research and investigate the application of entity linking by designing a novel profiling technique that allows to model a person (e.g., faculty) with her relevant Wikipedia entities. On the top of these new profiles, we design a new suite of algorithms (Cifariello et al., 2019) for the retrieval of experts that achieve better performance than previous solutions based on deep neural network (Van Gysel et al., 2016b). Complementary to the improvements achieved by our solution, we show in a public available system called Wiser, that the explicit modeling of people through Wikipedia entities helps users in the understanding and exploration of the research topics of individual faculties, as well as in supporting the activities of the Technology Transfer Office of our University when they are concerned with the search for research expertises that match some company or project needs. More details about Wiser will be given in Chapter 5.

1.3 Information Extraction

Information extraction (IE) is the task that aims at turning unstructured information residing in an input text into structured data (Jurafsky, 2000). Differently from knowledge extraction, information extraction is unconstrained in the sense that it does not require any specification schema or need of accessing to background knowledge (i.e., KG). Nevertheless, its extractions commonly serve as input for knowledge extraction tasks, such as entity linking (Shen et al., 2015; Usbeck et al., 2015), KG construction (Balasubramanian et al., 2012; Nguyen et al., 2017a; Shin et al., 2015) or question answering (Yao & Van Durme, 2014).

Example. One of the most popular information extraction's subtask is clearly the extraction of proper nouns mentioned in a text and label them with a type (i.e., named entity recognition, abb. NER) (Jurafsky, 2000). In the context of information extraction a *named* entity is an element that can generally be a person, place or organization, but it can also be a domain-specific element, such as a gene or a protein. For example, given the input text *"Leonardo is the scientist who painted Mona Lisa"*, a NER system recognizes *"Leonardo"* and

"Mona Lisa" as two persons, but without associating them to any meaning, i.e. the scientist `Leonardo_da_Vinci` and the portrait painting `Mona_Lisa`, respectively. Despite a named entity can possibly be associated to an element of KG, NER does *not* directly address the linking of named entities to KG's nodes.

Other information extraction's subtask examples involve coreference resolution (Clark & Manning, 2016), extraction of events (McClosky et al., 2011) and temporal expressions (Strötgen & Gertz, 2013; Verhagen et al., 2009), template filling (Chambers & Jurafsky, 2011) and open information extraction (Banko et al., 2007).

1.3.1 Open Information Extraction

Open information extraction (OIE) is the task that aims at extracting structured, machine-readable representation of information expressed in natural language text in an open-domain manner (Gashteovski et al., 2017). Extractions are relational phrase, commonly represented in the form of facts, i.e. subject-relation-object triples.

Example. Given an input text *"Leonardo is the scientist who painted Mona Lisa"*, an OIE system should correctly extract the two facts *("Leonardo", "is", "scientist")* and *("Leonardo", "painted", "Mona Lisa")*. Differently from knowledge extraction systems, OIE solutions do not associate any meaning to the extracted facts (e.g., *"Leonardo"* to `Leonardo_da_Vinci`).

The paradigm of OIE was introduced by (Banko et al., 2007) as a new technique for overcoming traditional IE approaches that usually need the human involvement in the form of hand-crafted rules or training examples, making their application unfeasible at Web-scale. The first OIE system proposed by (Banko et al., 2007) was TextRunner, and after that a large number of different proposals have appeared in literature and discussed below.

Open Information Extractors

TextRunner is the first OIE system that has been proposed in literature, together with the introduction of the open information extraction paradigm (Banko et al., 2007). TextRunner works in two stages. The first one is an off-line phase which aims at training a classifier through a self-supervision approach in order to learn a model that can label the extracted facts with a confidence label. This stage automatically generates its training data by extracting high-quality facts by means of several heuristics applied on extracted dependency trees. The second stage is the on-line phase that, given an input document, extracts facts relevant for its topics. More precisely, it performs a single pass on the input text by tagging sentences with a part-of-speech tagger and a chunker. Then, for each pair of noun phrases that are not too far, TextRunner determines whether or not to return the fact through the use of the classifier trained in the previous stage.

Woe is a system proposed by Wu & Weld (2010) as an improvement of (Banko et al., 2007). woe (which name stands for Wikipedia-based Open IE) modifies the first stage of TexTRunner in the construction of the training examples. Specifically, woe heuristically matches Wikipedia infobox values and the corresponding facts extracted in the Wikipedia article. woe is proposed in two versions. The first one that uses the same POS-based features of TexTRunner and the second one, whose features are based on dependency trees. Both proposals showed improvements with respect to TexRunner, with the POS-based version less accurate but significantly faster than the one based on dependency trees.

ReVerb (Fader et al., 2011) was designed in order to overcome the errors generated by state-of-the-art OIE systems, which commonly incurred in the extraction of uninformative and incoherent facts. ReVerb (Fader et al., 2011) solves these problems by implementing two rules that respectively use POS tag patterns expressed as regular expression and a dictionary of relations previously extracted from a large corpus collection by means of several heuristics. ReVerb performance showed significant improvements against both TextRunner and woe (both POS- and dependency tree-based versions).

Ollie. (Schmitz et al., 2012) enhanced ReVerb with a new system by proposing a new technique for learning relation-independent dependency tree patterns. Ollie (Open Language Learning for information extraction) takes inspiration from TextRunner by working in two stages but with several differences. The first stage (the off-line learning phase) runs ReVerb on the ClueWeb dataset[*] for automatically creating its training set as a set of sentences and the corresponding extracted facts. Then, a mapping between a path in the dependency

[*]lemurproject.org/clueweb09.php

tree to a fact is learned in order to encode the ways in which a fact may be expressed in a sentence. The second stage (on-line query phase) extracts facts from an unseen input text by means of several heuristics combined with the model learned in the first stage.

OpenIE. All previous described OIE systems (Banko et al., 2007; Fader et al., 2011; Schmitz et al., 2012; Wu & Weld, 2010) have been developed at University of Washington. In the last years they further improved their final extractor (Mausam, 2016), by making it more efficient and extended to n-ary facts. All these progresses have been made publicly available by their authors*.

ClausIE is a fully unsupervised OIE system that differs from the systems available in literature because it does not need of any labeled or unlabeled training data. ClausIE (Del Corro & Gemulla, 2013) runs a dependency parser on the input sentence and then it extracts facts by means of several principled heuristics designed by exploiting the properties of the English language. Given a dependency tree, the fact extraction is also very fast since it is implemented as a decision tree. ClausIE is also publicly available†.

MinIE. (Gashteovski et al., 2017) proposes a new OIE system in order to address and trade-off facts' compactness and accuracy performance. Specifically, MinIE is built on the top of ClausIE, by post-processing and enriching its facts in order to make them more informative via a set of properly designed attributes (such as polarity, modality or quantities). Most important, MinIE provides the possibility to compress facts by means of several heuristics of different level of aggressiveness:

- *Complete* mode remove extractions that contain subordinate facts.
- *Safe* mode prunes words from facts' constituent that are considered safe to drop, e.g. determiners, possessive pronouns, some adverbs and adjectives.
- *Dictionary* mode runs *safe* mode and then it searches for noun phrases matched by a regular expression. From the matched phrases MinIE prunes words that do not occur

*github.com/allenai/openie-standalone
†mpi-inf.mpg.de/departments/databases-and-information-systems/software/clausie

in a precomputed dictionary of stable constituent. This dictionary is built by processing the corpus using safe mode and then including the most frequent subjects, relations and objects.

- *Aggressive* mode prunes all words that are not considered essential, i.e. adverbial, adjective, possessive, temporal modifier, prepositional attachments, quantities modifying nouns, auxiliary modifiers and all compound nouns with different NE type than head word.

In this thesis, we use this OIE system for the extraction of open facts in Chapter 4. The source code of MinIE is publicly available[*].

Applications

After the introduction of the OIE paradigm (Banko et al., 2007), a large number of open information extractors have been proposed (see Section 1.3.1), with a number of solutions that can be easily deployed out-of-the-box on a variety of input texts thus encouraging their application over different domains (Mausam, 2016):

Event Schemas Extraction. An event schema is a set of actors that play different roles in an event (e.g., perpetrator, victim and instrument in a bombing event) (Balasubramanian et al., 2013). Several works (Balasubramanian et al., 2013; Romadhony et al., 2016) have shown that the OIE paradigm can be effectively used for the fully automatic construction of open-domain event schemas, overcoming the coherence issues (i.e. unrelated events and actors mixed together) that afflicted previous approaches (Chambers & Jurafsky, 2009). Beyond the end-to-end construction of event schemas, open facts can be also used for clustering (Romadhony et al., 2016) and discovering of events (Balasubramanian et al., 2012), as well as for retrieving at query-time pertinent and related information for an input event (Balasubramanian et al., 2013).

Machine Comprehension. The clean structure of open facts can be also used for computing the similarity between words and sentences. Stanovsky et al. (2015) use open facts for

[*]github.com/rgemulla/minie

answering to a machine comprehension task (Richardson et al., 2013) by building an unsupervised lexical matching on the top of extracted facts. Stanovsky et al. (2015) also show that using facts as context for training word embeddings (Mikolov et al., 2013b) allows the learning of higher quality vectors (extrinsically evaluated on word similarity and analogy tasks) than standard approaches based on BoW.

Knowledge Graph Population. The direct use of extracted open facts for deriving KGs can lead to the creation of very noisy resources of knowledge. On the other hand, with a proper canonicalization of facts' constituents (Galárraga et al., 2014; Nakashole et al., 2012), OIE systems can be used for the population of KG by dynamically acquiring new information from different sources as timely and comprehensively as possible (Dong et al., 2014; Mitchell et al., 2018; Nguyen et al., 2017a).

Summarization. In the context of summarization (Christensen et al., 2013), open facts have been preliminary used for determining whether two sentences contain or not similar information and thus avoiding the introduction of redundant elements in the final summary. In this thesis, we do a step further, and we use facts as key elements for the computation of single-document summaries through the proposal of a new task that we call *fact salience*, which actually introduces the problem of detecting the *salient* facts within an input document. Furthermore, we design and implement SalIE, the first *salient* open information extractor to date. More details about fact salience and SalIE will be given in Chapter 4.

2
Tools

In this chapter we introduce a suite of tools that constitute the building blocks of the novel algorithmic techniques that we have designed and we present in the next chapters. The first part is devoted to graph-based algorithms that use random walks for, respectively, computing centrality or similarity scores of nodes in a graph. Then, the second part introduce word embeddings, the popular latent representation of words that is widely used as modern alternative to the bag-of-words paradigm for modeling an input text with a small and dense vector of floating-point numbers. These two tools are fundamental for the understanding of the results presented in this dissertation, since they are broadly used in all chapter of the thesis. The third part presents gradient tree boosting, a machine learning technique commonly used for solving large-scale supervised learning problems. Finally, in the last part of this chapter, we describe hierarchical density-based clustering, a novel and efficient clustering algorithm for partitioning data points into high-quality clusters without sacrificing the scalability performance.

All sections of this chapter are structured in order to first provide a general overview of the topic at hand and then a description of the particular algorithmic adaptations that are used in the thesis. Furthermore, at the end of each section, a paragraph called *Context of Usage* outlines the context on which the introduced tool(s) are used in the dissertation.

2.1 Algorithms Based on Random Walks

Since the birth of the first computer networks, there has always been an ever-increasing interest in capturing the notion of centrality (aka relevance, importance or prominence) of the nodes in a given graph (Wasserman & Faust, 1994). Despite we can informally infer what an important node is, literature does not offer a formal consensus on its definition (Boldi & Vigna, 2014). For example, a node can be considered relevant if it has an high degree, if it is the closest to the others or if it has a large number of shortest paths that pass through it. These observations actually lead to the corresponding views of node centrality: namely, degree, closeness (Bavelas, 1948) and betweenness (Anthonisse, 1971).

PageRank. Among the number of algorithms proposed for computing the importance of the nodes in a graph (Boldi & Vigna, 2014), the leading representative is PageRank (Page et al., 1999), whose popularity is clearly related to its alleged use in the Google search engine (Brin & Page, 1998). PageRank lies under the category of methods known as *spectral algorithms*, which actually aim at computing the left dominant eigenvector of a matrix properly derived from the graph. Technically speaking, PageRank implements the following definition:

$$p^{(k)} = dA^T p^{(k-1)} + (1-d)p^{(0)} \tag{2.1}$$

where $p^{(0)}$ is the personalized (or teleport) vector, A is the adjacency matrix (row normalized) that models the edges between the nodes of the input graph and $d \in (0,1)$ is the damping factor, which incorporates in the algorithm the probability of jumping from a given node to another random node. Intuitively, PageRank models a random surfer that walks over the graph by starting at node u and following the edges with probability d or restarting the walking with probability $(1-d)$. After k iterations, the centrality score of a node u is given by $p^{(k)}(u)$ and d is usually chosen as a value close to 1 (e.g., 0.85), whereas $p^{(0)}$ can be instantiated in the most convenient way with respect to a given domain of interest. More precisely, the classical PageRank (Page et al., 1999) is run with $p^{(0)}$ set as uniform vector, whereas its personalized vector (e.g., giving a higher probability to a specific subset of nodes) results into that algorithmic variant commonly known as Personalized PageRank (PPR) (Haveliwala, 2002).

CoSimRank. Beyond the computation of the centrality scores of a graph, random walks can be also used for computing the similarity between nodes. The intuition of using random walks for deriving how much two nodes are similar relies on the idea that starting random walks from similar nodes will result into similar paths.

In this context, Rothe & Schütze (2014) proposed CoSimRank, an unsupervised graph-theoretic similarity measure whose definition hinges upon a combination of the vectors $p^{(k)}$ derived by the PPR algorithm starting from the compared nodes. More precisely, given two query nodes u and v, CoSimRank runs two different instances of PPR, by properly instantiating the personalized vectors with their standard basis $p_u^{(0)} = e_u$ and $p_v^{(0)} = e_u$, respectively. The element e_u (resp. e_v) is the vector with the u-th (resp. v-th) entry set to 1 and all other entries to 0. After running these two instances of PPR, the similarity score s between u and v is eventually computed by taking advantage of the centrality vectors of u and v that are derived at each iteration of PPR:

$$s(u, v) = \sum_{k=0} c^k \cdot \text{cosine}(p^{(k)}(u), p^{(k)}(v)) \tag{2.2}$$

where $p^{(k)}(u)$ (resp. $p^{(k)}(v)$) is the centrality vector of node u (resp. v) after k iterations, cosine is the operator of cosine similarity between two given vectors and $c \in (0, 1)$ is the decay factor introduced for weighting more early meetings than the later ones.

Context of Usage. In this dissertation PageRank and CoSimRank are used in Chapter 3 for estimating the relatedness between Wikipedia entities. Furthermore, in the remaining chapters, PageRank is used as leading technique for estimating the relevance of nodes in a graph in a fully unsupervised fashion. More precisely, the graphs on which we run PageRank differ in the type of the nodes (i.e., entities or facts), the weights of the edges (i.e., computed with different relatedness algorithms) and for the instantiation of the teleport vector (i.e., uniform vs positional vs frequency-based).

2.2 Word Embeddings

Traditional NLP and IR solutions model the input text as a discrete set of atomic units, where each word is identified with the index within a specified vocabulary (i.e., bag-of-words). Despite its popularity, this model has several weaknesses. For example, the semantics between words is completely ignored: according to this representation, the words *"Einstein"*, *"Relativity"* and *"Cat"* are equally distant, though *"Einstein"* is more semantically related with *"Relativity"* than to *"Cat"*. Second, this technique produces a very sparse model with a consequent large size of vocabulary and their vector representations (course of dimensionality).

Following the widespread application of deep learning, Mikolov et al. (2013b) proposed Word2Vec, a novel technique that aims at solving these issues through the use of a language model learned by a neural network (Bengio et al., 2003). More precisely, Word2Vec algorithms efficiently learn high-quality and low-dimensional vectors that allow to encode semantically similar words into similar vectors. Surprisingly enough, these vectors can also be used to derive representations of other words via linear translations, for example:

$$\textit{vector(``King'') - vector(``Man'') + vector(``Woman'')} \approx \textit{vector(``Queen'')}$$

which means that *vector("King") - vector("Man") + vector("Woman")* gives the vector closest to *vector("Queen")*.

The vector representation of words (embeddings) proposed by (Mikolov et al., 2013b) is actually the matrix between the hidden and either input or output layer of a neural network learned after a training phase. The neural network at hand can be trained in two different ways in order to produce two distinct language models. Specifically, the continuous bag-of-words (CBOW) trains the neural network in order to learn a model capable of predicting a word given an input context (e.g., sequence of words), whereas Skip-gram trains the neural network in order to predict the context given an input word. Both CBOW and Skip-gram models are designed on the top of a fully-connected neural network with one single hidden layer and trained with the stochastic gradient descent learning algorithm (Mikolov et al., 2013a).

Literature currently abounds of research on word embeddings as well as surveys (Bakarov, 2018; Li & Yang, 2018; Simov et al., 2017; Wang et al., 2018) that attempt to summarize the large amount of work that is done every year for learning higher quality representations of words and phrases (Bojanowski et al., 2016; Joulin et al., 2016; Lample et al., 2017; Le & Mikolov, 2014; Levy & Goldberg, 2014a; Luong et al., 2013). Among them, we mention GloVe, the method proposed by Pennington et al. (2014), which has shown how Word2Vec can be derived through a factorization method applied on a word-word co-occurrence matrix.

Entity2Vec (Ni et al., 2016) is a recent extension of the original Word2Vec model that easily allows to generate the embedding representation of Wikipedia entities. Entity2Vec works in two steps. First, it preprocesses the Wikipedia corpus by substituting each internal Wikipedia hyperlink with a proper placeholder. For example, supposing that the Wikipedia ID of `Leonardo_da_Vinci` and `Mona_Lisa` are 18079 and 70889, respectively, the preprocessing step will transform the html text:

```
<a href="wiki/Leonardo_da_Vinci"> Leonardo </a>
painted
<a href="wiki/Mona_Lisa"> Mona Lisa </a>
```

into:

```
entity_18079 painted entity_70889
```

Second, the Word2Vec algorithms are run on this preprocessed Wikipedia corpus in order to learn the embedding representation of words and entities into the same latent space. Finally, the embedding vector of entity `Leonardo_da_Vinci` (with ID 18079) can be retrieved by querying the index with `entity_18079`.

DeepWalk (Perozzi et al., 2014) is another recent extension of the original Word2Vec developed for learning the latent representations of nodes in a graph. DeepWalk works in two steps. First, given an input graph $G = (V, E)$, a set of k short truncated random walks are generated for each node $u \in V$. Second, the $k \cdot |V|$ generated random walks are used to feed the Word2Vec algorithms for learning the embedding representation of the nodes, here interpreted as “words” in “sentences” formed by the nodes traversed by those random walks. The general intuition is that similar nodes should actually generate similar paths and thus embed to similar vectors.

Context of Usage. Entity2Vec and DeepWalk are used for learning different latent representations of entities, by respectively leveraging the Wikipedia corpus and its graph structure. Specifically, in Chapter 3 and 5 we use them as techniques for computing the relatedness between Wikipedia entities, while in Chapter 4 they are used as tools on which we design a novel set of fundamental features for the detection of salient entities in text. Moreover, in Chapter 4 we use GloVe as technique for learning the embedding of words and consequently use it for computing the relatedness between open facts within our novel framework SALIE.

2.3 Gradient Tree Boosting

Among the great number of methods proposed in the literature (Michalski et al., 2013) for solving regression or classification problems, gradient tree boosting (GTB) (Chen & Guestrin, 2016; Friedman, 2001) is clearly one of the most widely and highly effective adopted solutions. As opposite to complex models, such as neural networks, where the model is first built and then fitted upon its hyper-parameters, boosting techniques start from a very simple model and then they incrementally grow their complexity through an ensemble of weak learners that is expanded at every iteration of the training phase. In the context of GTB, weak learners are regression trees (Breiman et al., 1993), but they can easily be adapted for classification problems by converting the output scores into probability with sigmoid or softmax functions. Given an input example (e.g., vector of features) x, the prediction score of a GTB is computed as a majority voting among the individual weak learners:

$$F(x) = \sum_{m=1}^{M} h_m(x) \tag{2.3}$$

where h_m is the function associated with the m-th regression tree (i.e., a weak learner). At iteration i of the training phase, GTB produces an ensemble F_{i+1} of weak learners, constituted by F_m and the weak learner that compensates the shortcomings of F_{m-1}. Technically speaking, at the m-th iteration, GTB chooses the regression tree h_m that minimizes the loss function L (e.g., least squares):

$$h_m = \underset{h}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + h(x_i)) \tag{2.4}$$

where N is the size of the training set and y_i is the ground-truth score associated to x_i. GTB models Equation 2.4 as an optimization problem that can be solved via gradient descent: at the m-th iteration, the boosting algorithm computes the gradient of the loss function evaluated in F_m and then it chooses the regression trees h_m that minimizes its value.

Context of Usage. In Chapter 4 we use XGBoost (Chen & Guestrin, 2016) as one of the main components of Swat, the state-of-the-art system that we propose for detecting the salient Wikipedia entities of an input document. More precisely, XGBoost is an implementation of GTB properly enhanced with a variegate set of sparsity- and cache-aware algorithms that have been specifically designed for making it highly scalable both on local and distributed settings over large-scale datasets.

2.4 Hierarchical Density-Based Clustering

Clustering addresses the problem of identifying groups, or clusters, of data points with similar characteristics (Bishop, 2016). Since the "similarity" between the points of a data collection is commonly calculated as a distance in the multidimensional space at hand, a cluster is intuitively defined as a set of data points whose inter-distances are small compared to the intra-distances outside of the cluster.

Literature offers a number of solutions for solving this problem (Xu & Wunsch, 2005), with a broad suite of techniques that achieved a widespread popularity in the offering a plethora of performances in terms of efficiency and efficacy. Leading examples are k-means (Lloyd, 1982), DBSCAN (Ester et al., 1996) and hierachical clustering (Rousseeuw & Kaufman, 1990), which have been complemented by several extensions, such as x-means (Pelleg et al., 2000) and HDBSCAN (McInnes & Healy, 2017).

In this section we provide an overview of HDBSCAN, one of the last proposals and publicly available[*] clustering algorithms. HDBSCAN combines hierarchical and density-based techniques into an unified approach that has shown to generate higher-quality clusters than other popular algorithms without sacrificing the scalability performance (McInnes & Healy, 2017).

[*]github.com/scikit-learn-contrib/hdbscan

We now first recall several basic terminologies of DBSCAN and then we move into its hierarchical adaptation called as HDBSCAN (McInnes & Healy, 2017).

DBSCAN. The input of every clustering algorithm is a metric space constituted by a set of data points $X = \{X_1, X_2, \dots, X_n\}$. DBSCAN needs two more parameters to be executed: ε, the distance scale, and k, the density threshold expressed in terms of a minimum number of points. A point X_i is a *core point* if $B(X_i, \varepsilon)$, the ball of radius ε pivoted in X_i, contains at least k points. We say that X_i and X_j are *density-connected* if they are both core points and they are contained within each others balls, i.e. $X_i \in B(X_j, \varepsilon)$ and $X_j \in B(X_i, \varepsilon)$. A cluster is eventually determined as a non-empty maximal subset of points which are density connected.

HDBSCAN (McInnes & Healy, 2017) grows a hierarchy of DBSCAN clusterings for different values of ε. The algorithm starts by building the tree from the root and then it proceeds downward. Differently from DBSCAN, HDBSCAN is based on the notion of density expressed as $\lambda = 1/\varepsilon$. At each cluster split we consider the child clusters that can fall into three distinct cases:

1. A cluster C_i contains less than m points. In this case, C_i is considered *spurious* and all its data points are marked as *outliers*.

2. C_i is the only cluster with more than m points. C_i is considered as continuation of its parent and no more split are performed.

3. More than a single child cluster contains m points. This case is considered a "true" split since it is shrinking the size of the parent cluster and thus increasing the density λ within the new generated clusters.

After the execution of this hierarchical algorithm, HDBSCAN eventually outputs the flat clustering that maximizes the sum of the stability function σ expressed as:

$$\sigma(C_i) = \sum_{X_j \in C_i} (\lambda_{max,C_i}(X_j) - \lambda_{min,C_i}(X_j)) \tag{2.5}$$

where $\lambda_{max,C_i}(X_j)$ is the λ value for which X_j falls out of the cluster C_i and $\lambda_{min,C_i}(X_j)$ is the minimum λ for which X_j is present in C_i.

Context of Usage. In Chapter 5, where we model the expertise of people through a graph of Wikipedia entities, HDBSCAN is used within an outlier-elimination process for recognizing and removing from those entities the ones that do not belong to any cluster and thus may be considered off-topic.

Part II
Knowledge and Information Extraction

3 Algorithms for Computing the Relatedness between Entities

MEASURING HOW MUCH TWO NODES in a knowledge graph are related is a core functionality of a large number of knowledge extraction tasks. Unfortunately, the comparison between relatedness measures has commonly been performed through their use in downstream applications, while their *intrinsic* evaluation has been rare and confined mainly to word pairs. Towards deeper understanding of all well-known relatedness measures, we introduce in this chapter a new dataset with human judgments of entity relatedness, complemented with a thorough study of all entity relatedness measures proposed in recent literature. We then propose a new, space-efficient and 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. This framework shows better agreement with human judgment than existing proposals both on the new dataset and on two established ones. We also plug our relatedness algorithm into a state-of-the-art entity linker and observe an increase in its accuracy and robustness.

3.1 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., (Bordino et al., 2013; Ni et al., 2016; Scaiella et al., 2012) — to short posts, news and queries — see, e.g., (Blanco et al., 2015; Cornolti et al., 2016; Meij et al., 2012). 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 (Usbeck et al., 2015). In contrast, *intrinsic* evaluation against human judgments of relatedness has been rare and confined mainly to word pairs — see, e.g., (Agirre et al., 2009; Gabrilovich & Markovitch, 2007; Milne & Witten, 2008; Radinsky et al., 2011) — 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 in this chapter WiRe, a new dataset of 503 pairs of entities occurring in Wikipedia and drawn from the New York Times dataset (Dunietz & Gillick, 2014) with human-assigned relatedness scores. This is in contrast to *word* similarity datasets such as WikiSim (Milne & Witten, 2008), 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 chapter 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 & Witten, 2008), widely used in the best entity linkers (Cornolti et al., 2016; Cucerzan, 2007; Ferragina & Scaiella, 2012; Kulkarni et al., 2009; Shen et al., 2015; Usbeck et al., 2015). 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.

We supplement the above intrinsic evaluation with an extrinsic evaluation in the domain of entity linking. We plug our newly proposed relatedness framework into a popular and state-of-the-art entity linker, TagMe (Ferragina & Scaiella, 2012). There is a clear increase in entity linking accuracy and in addition, the entity linker becomes less sensitive to one of its critical system parameters.

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. Comprehensive evaluation of relatedness measures proposed in recent literature over the new WiRe and the known WikiSim datasets.
3. A new, two-stage, fast and space-efficient entity relatedness framework that returns more accurate scores than prior proposals, as per intrinsic tests. Specifically, the improvement is more than 5%, with peaks of 7% on WiRe. We also discuss a compressed version of our framework that fits in few hundreds MBs still guaranteeing time efficiency at the same accuracy performance.
4. Evaluation of our approach in the context of ranking pairs of entities over the Kore dataset, with improvements with respect to the state-of-the-art that span from 2.9% to 18.8%.
5. Extrinsic evaluation of the new proposal over the entity linker TagMe shows benefits in accuracy and robustness.
6. Publicly available WiRe data and code of all tested algorithms.

3.2 Related Work

Recent literature offers many algorithms for estimating relatedness of words in the WordNet graph (see Agirre et al. (2009) 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. 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 chapter 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. They are reviewed in detail in Section 3.3. Here we mention that, among other algorithms, we will test the best-known and most popular ones such as ESA (Gabrilovich & Markovitch, 2007), the method proposed by Milne & Witten (2008), and the most recent approaches based on entity embeddings (Ni et al., 2016).

The literature offers other approaches to entity relatedness which are either *supervised* and applied to learning-to-rank frameworks (Ceccarelli et al., 2013), or deploy external/additional sources of information such as temporal, categorical, or crawled (Web) documents (Agirre et al., 2009; Radinsky et al., 2011; Strube & Ponzetto, 2006). The comparison with these results is out of the scope of this work.

3.2.1 Terminology

In this section we specialize the terminology originally introduced in Section 1.1.3. More precisely, we will use $I(u)$ and $O(u)$ to denote the in- and out-neighbors of the node u in the Wikipedia KG, respectively, and denote by $\Gamma(u) = I(u) \cup O(u)$ the undirected neighborhood of u. In the following the *distance* between nodes is measured as unweighted length of the shortest path in the undirected KG.

3.3 Known Relatedness Methods

In this section we review the 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 chapter: document annotation (Piccinno & Ferragina, 2014), word and document similarity (Gabrilovich & Markovitch, 2007), personalized Web search (Haveliwala, 2002), machine translation (Rothe & Schütze, 2014), document classification (Perozzi et al., 2014; Tang et al., 2015), and link prediction (Liben-Nowell & Kleinberg, 2007). 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, and is described in Section 3.3.1. The second one includes methods which use the hyperlinked structure of the graph connecting the entities, and it is described in Section 3.3.2.

3.3.1 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 (entity definition) document 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 3.6.

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 cosine between their corresponding vectors.

Explicit Semantic Analysis (ESA) (Gabrilovich & Markovitch, 2007). 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) (Pauls & Klein, 2011). 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 & Klein (2011) for details.

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 chapter 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) (Hoffman et al., 2010). 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 the input corpus, LDA can map documents to a vector of weights over these latent topics. Entities are compared by computing the cosine between the topic distribution weights of their definition documents.

Entity2Vec (Ni et al., 2016). This is the application of Word2Vec (Mikolov et al., 2013b) 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.

Table 3.1: 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	$\lvert\Gamma(u) \cap \Gamma(v)\rvert \,/\, \min\{\lvert\Gamma(u)\rvert, \lvert\Gamma(v)\rvert\}$

3.3.2 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 3.1 reports the ones based on immediate neighbors.

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 (Haveliwala, 2002). 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. (2014)).

CoSimRank (Rothe & Schütze, 2014). This is a recent enhancement to PPR+Cos introduced to find synonyms and translations of words. It combines features of SimRank (Jeh & Widom, 2002) 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 (Fogaras & Rácz, 2005; Jeh & Widom, 2002) because CoSimRank was established to be better in terms of time/space complexity and accuracy (Rothe & Schütze, 2014).

WikiWalk (Yeh et al., 2009). This hybrid approach applies PPR+Cos while using the ESA vector of nodes to bias the teleportation jump of the random walk.

Commute Time (F. Fouss & Saerens, 2005). 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.

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 (Perozzi et al., 2014). Therefore, in our experiments, we will omit SVD, LINE and Commute Time.

DeepWalk (Perozzi et al., 2014). 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 (Tang et al., 2015). 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) (Golub & Reinsch, 1970). Levy & Goldberg (2014b) 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.

3.4 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 (Milne & Witten, 2008) 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*[*] 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).

3.4.1 Phase 1: Generation of Many Entity Pairs

We drew entity pairs from the dataset published by Dunietz & Gillick (2014), which has 2 203 909 entity annotations within about 100 000 news articles drawn from the New York

[*]In this chapter, the distance between nodes is measured as the unweighted length of the shortest path in the undirected KG.

Times Corpus (NYTC). An annotation is a tuple $(\textit{entity-ID}, \textit{news-ID}, \textit{salience})$, 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 (Dunietz & Gillick, 2014).

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. 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.

3.4.2 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, Organisation 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).

3.4.3 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 *indepen-*

dently 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 3.2, 3.3, 3.4 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 3.6.9) that WiRe's intrinsic evaluation will correlate well with extrinsic applications.

3.5 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 with increasing efficiency and efficacy. The experiments of Section 3.6 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.

3.5.1 First Stage: Subgraph Creation

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

Table 3.2: 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 3.3: WiRe dataset statistics bucketed according to the type of entity.

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

Table 3.4: 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

Entity2Vec) and the best methods that deploy its hyperlinked structure (i.e., DeepWalk or Milne&Witten). The nodes of the subgraph $G_C = (V, E)$ will be defined as $V = R_u \cup R_v \cup \{u, v\}$. The value of k will be set to 30 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: Creating Edges of Subgraph G_C. The edge set E is defined as a sparse graph consisting of $|E| = \mathcal{O}(k)$ undirected edges which connect every node in $R_u \cup R_v$ to both u and v. 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 3.6.7), and we will investigate the impact of three main edge-scores: Milne&Witten, Entity2Vec and DeepWalk, selected as a result of the large experimental analysis conducted in section 3.6.

3.5.2 Second Stage: Computing Relatedness

The relatedness score between u, v is derived by computing the CoSimRank 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.

Discussion

Section 3.6 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 three key reasons:

1. Edges in Wikipedia (or any typical KG) are only a noisy hint of relatedness. Edges may be introduced by authors for different "goals": citation, elaboration, explanation, etc. Moreover, different communities of Wikipedia pages offer different "densities" of hyperlink annotations.

2. The nodes of the subgraph G_C are strongly related to the queried nodes u and v in terms either of textual content or neighbor structure, hence they constitute good potential "bridges" for establishing their relatedness.
3. The edges of the subgraph G_C are more *meaningful* and *less noisy* than the edges present in Wikipedia because they are confined to few "meaningful bridge nodes".

These intuitions will be supported by the wide set of experiments reported in Section 3.6.

3.6 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 to entity linking using our two-stage framework. Datasets and all algorithms we experimented have been publicly released.[*]

3.6.1 Datasets

The first dataset we use is the one introduced in this chapter, 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 manually mapped into their corresponding Wikipedia entities by Milne & Witten (2008). We further filtered WikiSim by (counts of discarded pairs are given in parentheses): 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).

Figure 3.2 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 confidence that accurate relatedness estimates can generalize across datasets.

[*]github.com/mponza/WikipediaRelatedness

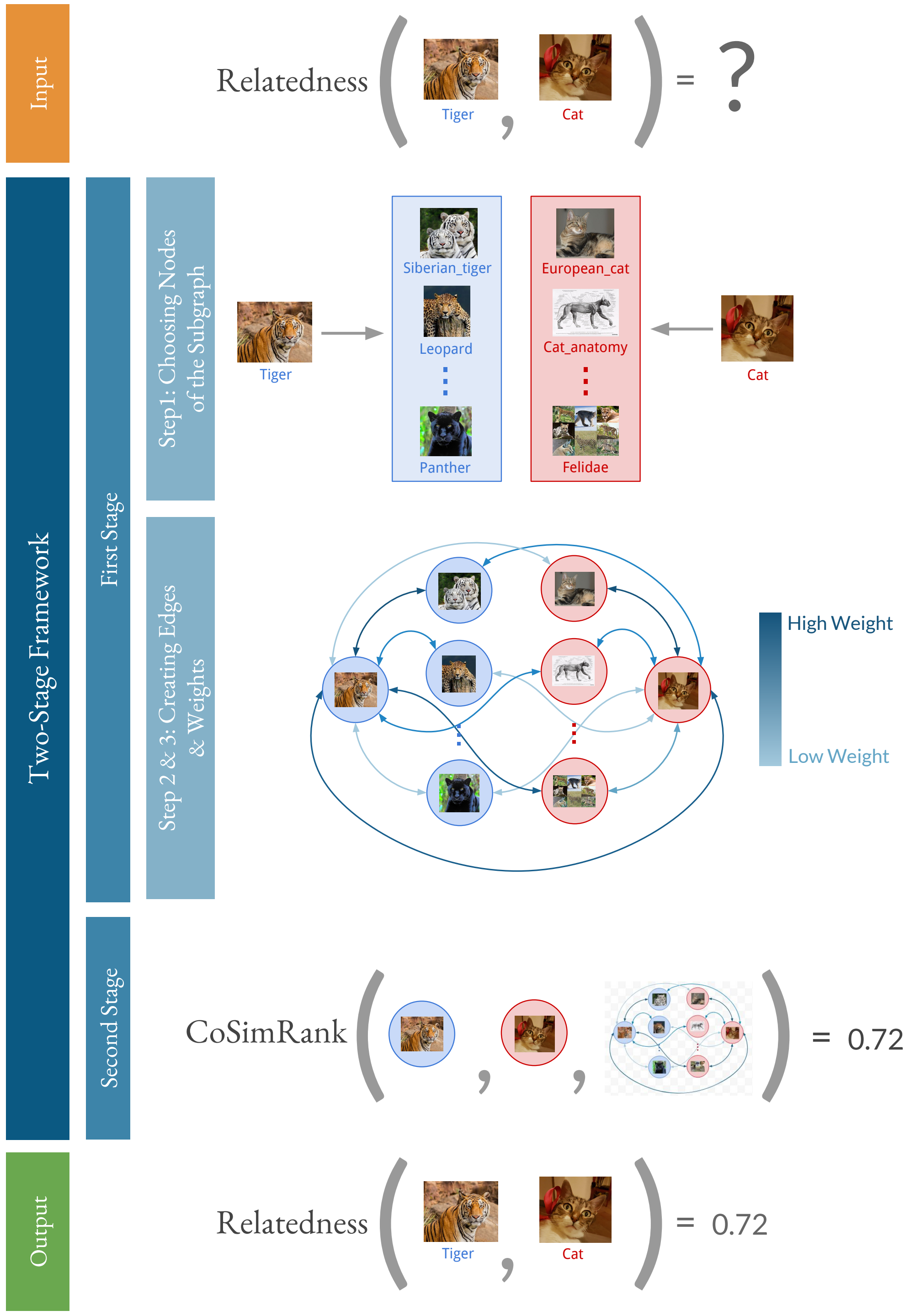


Figure 3.1: Example of the two-stage framework on two query entities `Tiger` and `Cat`.

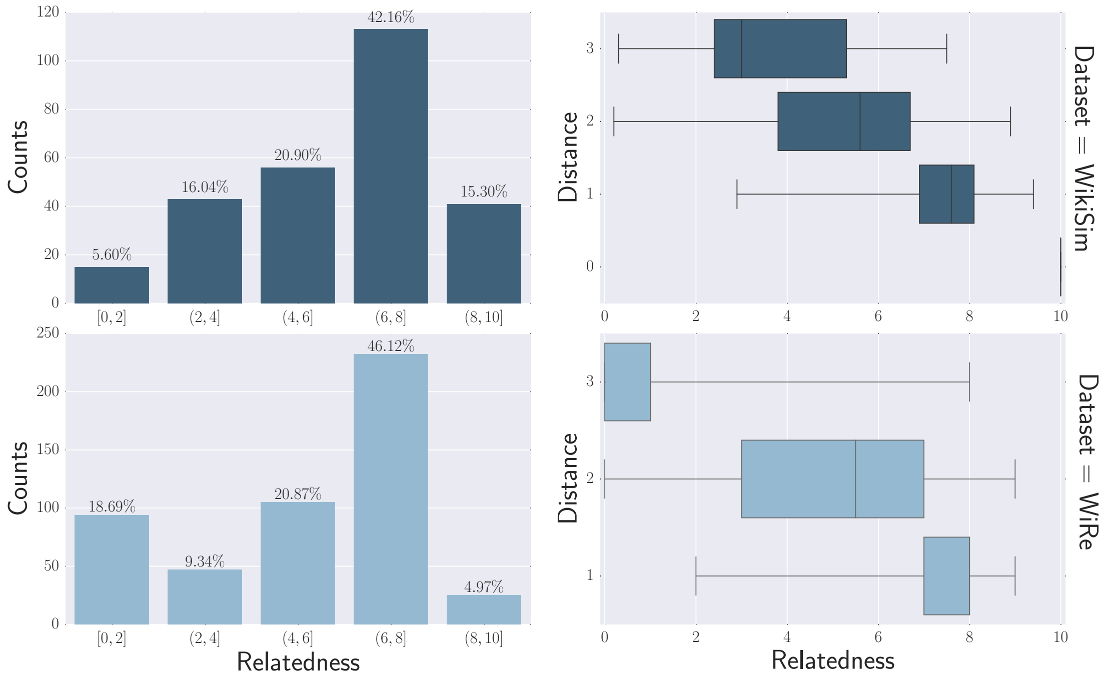


Figure 3.2: Data characteristics (top: WikiSim, bottom: WiRe). Charts in the left column show counts of instance pairs with relatedness in bucketed ranges. Most pairs are moderately related. Charts in the right column show distance distribution against bucketed relatedness. Generally relatedness reduces with graph distance, but there is a broad spread.

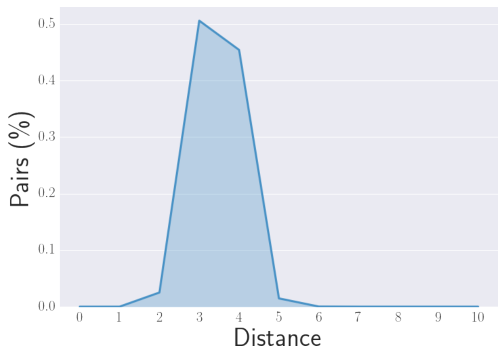
Significantly (right column in Figure 3.2) many entity pairs are similar (resp., dissimilar) yet far (resp., near) in the graph. Therefore, effective relatedness formulations cannot assume that KG distance between entities is a *good predictor* of their relatedness.

For the sake of completeness we mention that the datasets developed by Ceccarelli et al. (2013); Hoffart et al. (2012) 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.

3.6.2 System Details

We give a few technical details about the implementation and the settings of the methods described in Section 3.3.

Wikipedia as Knowledge Graph. We used the Wikipedia dump of March 2016. The Wikipedia corpus was normalized by removing punctuation elements and lowercasing words.



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

Figure 3.3: Probability mass function of the distance distribution (left) and several statistics (right) computed on the Wikipedia graph.

The Wikipedia graph was indexed with WebGraph (Boldi & Vigna, 2004) deriving the statistics reported in Figure 3.3. 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 & Markovitch (2007) and Ni et al. (2016) because these software are not available. Wikipedia corpus was lemmatized with CoreNLP[*], stopwords removed, and indexed by Lucene[†]. 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 (Hoffman et al., 2010) 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[‡]. To obtain Entity2Vec embeddings and LM probabilities, we replaced outbound hyperlinks to Wikipedia pages with a unique placeholder token (Ni et al., 2016), and processed this corpus using Word2Vec and BerkeleyLM (Pauls & Klein, 2011) respectively. In DeepWalk we set the transition probability from u to v as proportional to the frequency of a link to v within the textual description of u.

[*] stanfordnlp.github.io/CoreNLP
[†] lucene.apache.org
[‡] radimrehurek.com/gensim

For the algorithms CoSimRank* and DeepWalk† 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.

3.6.3 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 (Mohler & Mihalcea, 2009; Strube & Ponzetto, 2006), and the **Spearman** correlation index (Gabrilovich & Markovitch, 2007; Yeh et al., 2009). The Pearson index focuses on the difference between predicted-vs-correct relatedness *scores*. In contrast, the Spearman index focuses on the *ranking order* among 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 (Cornolti et al., 2016; Ferragina & Scaiella, 2012; Kulkarni et al., 2009; Ni et al., 2016). Spearman's index is crucial when the *correct ranking* among entity pairs based on their relatedness is required, as in other entity linkers (Cucerzan, 2007; Piccinno & Ferragina, 2014).

Therefore, we follow Hassan & Mihalcea (2011) 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 (Ganea et al., 2016; Usbeck et al., 2015; Zwicklbauer et al., 2016)). 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 the statistical significance of our results with the methodology described by Radinsky et al. (2011).

*github.com/casaro/CoSimRank
†github.com/phanein/deepwalk

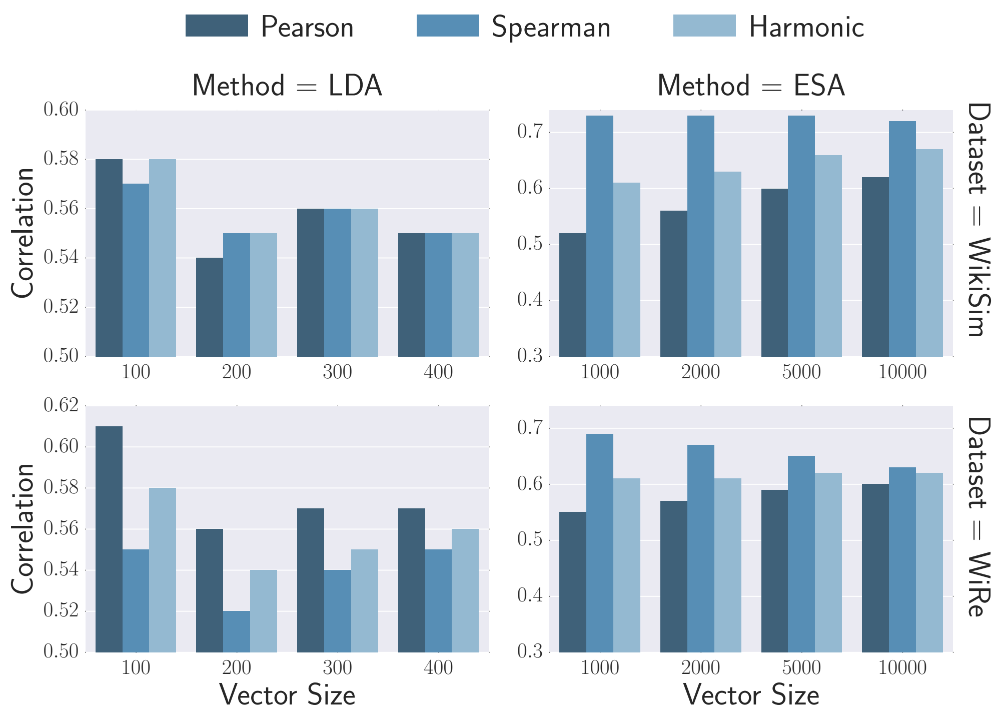


Figure 3.4: Relatedness performance over WikiSim (top) and WiRe (bottom) datasets by ranging respectively the size of LDA and ESA vectors.

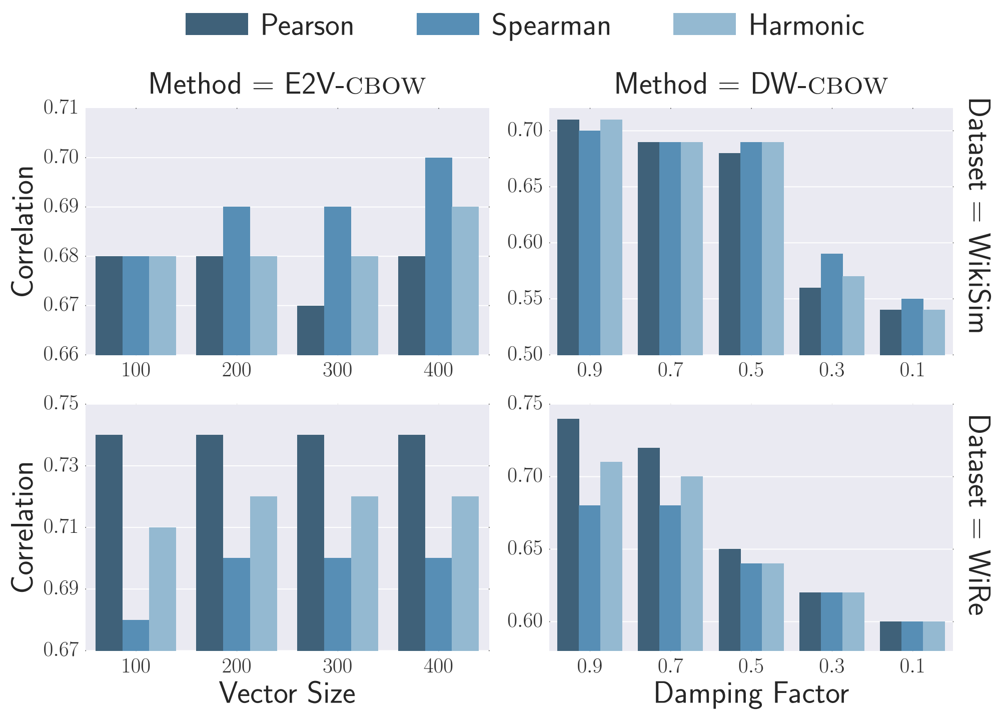


Figure 3.5: Relatedness performance over WikiSim (top) and WiRe (bottom) datasets by ranging respectively the size of LDA and ESA vectors .

3.6.4 Comparative Evaluation of Building Blocks

Toward selecting the best systems for our two-stage framework, we first compared all prior formulations and their various hyperparameters. We evaluated different lengths of ESA's concept vectors between $[100, 10\,000]$. We tried different lengths for the vectors built by LDA and Entity2Vec, between $[100, 400]$. We varied the damping factor of DeepWalk in $[0.1, 0.9]$. All results are reported in Figures 3.4 and 3.5. 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 were chosen to produce Table 3.5. For each method surveyed in Section 3.3, we show the four performance indexes. From the upper part of Table 3.5, we notice that the methods based on text show consistent performance across Spearman and Pearson indexes over both datasets.

From the lower part of Table 3.5, 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 indices (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 (Usbeck et al., 2015) seems amply justified.

The top-performing methods over both 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. Since they involve the entire KG in their computation they are also very slow, so we will no longer consider them in the remaining experiments.

Table 3.5: Relatedness performance of the methods surveyed in sections 3.3.1 and 3.3.2. 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 Copuling	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

3.6.5 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:

- 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 *out*-neighbors of u and v according to Milne&Witten's

Table 3.6: Relatedness performance of our novel two-stage framework described in Section 3.5. 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 3.5 (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 3.5		0.71	0.72	0.71	0.77	0.76	0.72	0.710

score. (We experimented with in-, out- and all neighbors and out-neighbors gave the best results, reported in Table 3.7).

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.

3.6.6 Two-Stage Results and Analysis

Table 3.6 reports the best results obtained for each one of the configurations described above, where the last line helps to compare against the best performance *for each individual correlation index* in Table 3.5. 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 3.5, 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 3.6), 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 3.6.4, 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 3.6, “Best Results from Table 3.5”, 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 3.5. 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 queried pairs, which do indeed occur in our datasets (Figure 3.2) as well as in applications.

Table 3.7: Comparing the best two methods for Step 3 in Table 3.6 against their linear combination, having Step 1 set to Milne&Witten (over out-neighbors).

Step 3	**WikiSim**			**WiRe**			**AVG**
	Pearson	**Spearman**	**Harmonic**	**Pearson**	**Spearman**	**Harmonic**	
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 3.5	0.71	0.72	0.71	0.77	0.76	0.72	0.710
Best Results from Table 3.6	0.73	0.72	0.73	0.83	0.73	0.78	0.745

3.6.7 Further Improvements via Combinations

Encouraged by the results above, we took one further step and jointly deployed all the information provided by the KG: namely, the neighborhood of a node (i.e., Milne&Witten), the neural embedding of its random walks (i.e., DW-CBOW) and the textual content of the node labels (i.e., E2V-CBOW). So we configured our two-stage framework with Milne&Witten for Step 1, because this is the method that shows the best trade-off between efficacy (see Table 3.6) and efficiency (see above). For Step 3 we used a linear combination of the two most effective methods in Table 3.6 — namely, Milne&Witten and E2V-CBOW or DW-CBOW. Table 3.7 reports the best results achieved by each combination, obtained by equally weighting the two methods at hand. Again the last two lines are introduced for comparison and report the best performance *for each individual correlation index* in Table 3.5 and 3.6, respectively.

The improvement achieved by this combined scheme is again significant: with respect to the known methods of Table 3.5, its harmonic mean is +3% over WikiSim and +7% over WiRe (thus resulting in a +5% on AVG); with respect to our "uncombined" approach, the improvement in the harmonic mean is slight but consistent over the datasets.

For the sake of completeness we have also experimented the configuration with ESA for Step 1 and the same linear combinations above for Step 3. Not surprisingly, the performance of this configuration is slightly better than the one shown in Table 3.7 (just +1% on the Harmonic mean over WiRe), but this comes at the cost of larger running times.

In conclusion, given the results of Table 3.7, we suggest the configuration which deploys Milne&Witten for Step 1 and the combination Milne&Witten+DW-CBOW for Step 3. It is both effective and efficient, with the latter issue being addressed in the next section.

Table 3.8: Relatedness performance on the task of ranking entity pairs over the Kore dataset (Hoffart et al., 2012). The first five columns distinguish the performance among different entity's topics, whereas the last two columns summarize them by their average. The two-stage framework is here instantiated with the best configuration resulted from Section 3.6.7, namely Milne&Witten for Step 1 and the combination of Milne&Witten and DW-cbow 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
Two-Stage Framework	0.729	0.676	0.696	0.580	0.680	0.672	0.699

3.6.8 Evaluation on Ranking Entity Pairs

In this section we evaluate our novel two-stage framework in the domain of ranking pairs of Wikipedia entities (Hoffart et al., 2012). 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 (Hoffart et al., 2012), constituted by a total of 21 seed entities grouped in 5 different topics (i.e., IT Companies, Hollywood Celebreties, 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 (Hoffart et al., 2012). As baselines we use the Kore methods (Hoffart et al., 2012), a set of unsupervised corpus-based techniques for the ranking of entity pairs, which are currently the state-of-the-art on the Kore dataset. Unfortunately, 28 entities of the Kore dataset are not present in our KG, thus we remove them and we recompute the spearman correlation of the original Kore techniques* without taking into account the removed entities. More precisely, 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 3.8. Our two-stage framework obtains very good performance among all different entity's topics. More precisely, it achieves the highest scores in Hollywood Celebrities and TV Series topics, with improvements of +4.2% and +18.8% with

*We thank Johannes Hoffart for providing us the outputs of Kore methods.

respect to the second-highest methods KORE (original) and KORE (LSH-G), respectively. 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). Our method arrives third only once on the Video Games topic, with a difference with respect to the first one of -18.5%. This is not surprisingly 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 average performance (last two columns of Table 3.8), both per topic and among all seed entities, with improvements of $+2.9\%$ and $+4.4\%$ with respect to KORE (original).

3.6.9 EXTRINSIC EVALUATION ON TAGME

We supplement the evaluation above with an extrinsic evaluation on the strongly motivated entity linking task. In the public domain, TAGME (Ferragina & Scaiella, 2012) is among the most popular, open-source[*] 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 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. Figure 3.6 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 (Usbeck et al., 2015), our relatedness measure not only improves TAGME, but also makes it more insensitive to choices of ε.

[*]github.com/gammaliu/tagme

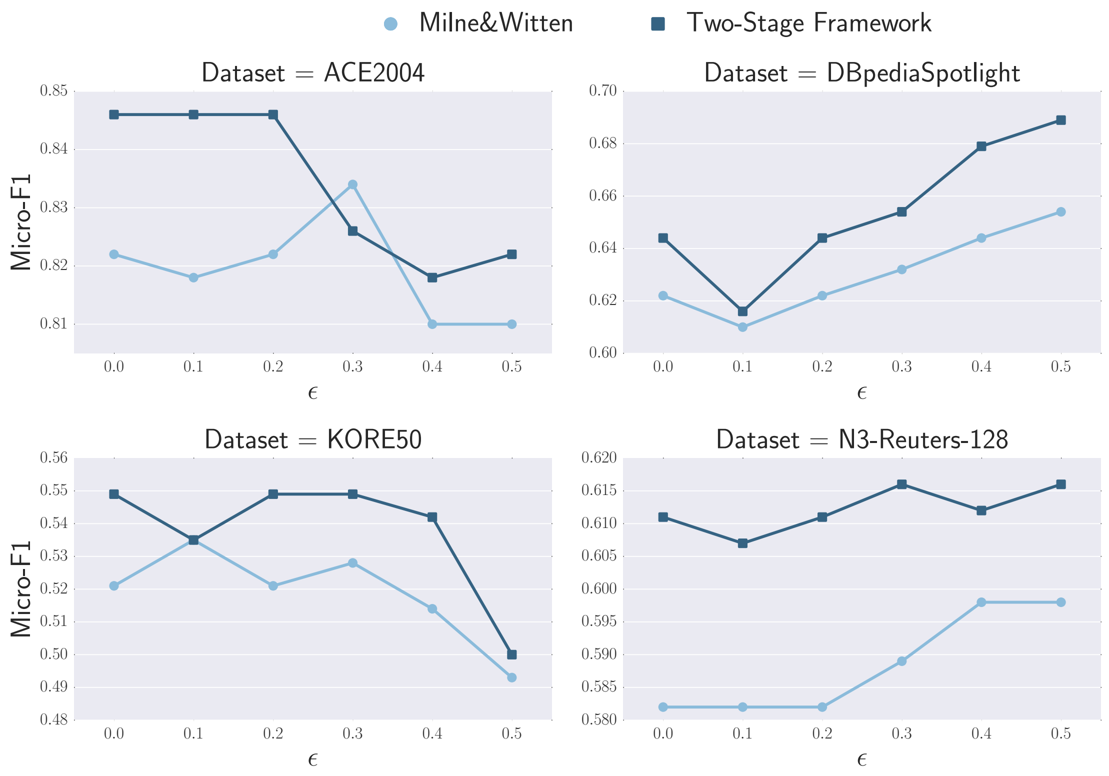


Figure 3.6: Entity linking performance by varying the ε-score of TagMe's voting scheme which deploys Milne&Witten and our two-stage framework, respectively, over four distinct text corpora.

3.6.10 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.

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:

Table 3.9: 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 3.10: Running time performance (milliseconds) of our two-stage framework configured as in Table 3.7.

		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

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[*] and it can be reduced via compression. We applied Elias-Fano codes from WebGraph (Boldi & Vigna, 2004) to compress the KG, and we used FEL (Blanco et al., 2015) to compress the embeddings. Table 3.9 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 ex-

[*]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 3.9, thus giving further advantage to the uncompressed space solution.

ecuted by applying CoSimRank over the graph G_C for just one single iteration (see Second Stage of section 3.5). 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 3.10 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 3x 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.

4 Algorithms for Entity and Fact Salience

Extracting salient information from an input document has become a fundamental task on which different downstream applications hinge upon to improve the quality of their results, such as Web search ranking (Luo et al., 2017), email tagging (Lahiri et al., 2017), contextual advertising (Anagnostopoulos et al., 2011) and news suggestion (Fetahu et al., 2015).
Automatic Document Summarization (Gambhir & Gupta, 2017) is the wide research area that concerns the extraction of salient information from an input document. More precisely, document summarizers aim at identifying relevant and topical information from an input document and condense them into a summar, i.e. a text which is shorter than the original one but that still preserves the salient elements that it conveys. Summaries are clearly fundamental from different point of views. They can enable fast and accurate search of documents from a large text collections (Hasan & Ng, 2014) as well as they can help a reader to immediately identify the relevant topics of the original document (Luhn, 1958).

In this chapter, we propose two different solutions for the automatic document summarization problem through the design and implementation of two novel systems for, respectively, the extraction of *salient Wikipedia entities*, and *salient open facts*. Specifically, in Section 4.1 we propose a new entity salience system, called Swat, which aims at producing a summary of an input document constituted of salient Wikipedia entities. Technically speaking, Swat lies into the subdomain of knowledge extraction called *entity salience*, which actually involves the design of solutions that pursue the objective of summarizing the content of an input document through the proper detection of *salient entities* (belonging to a knowledge graph) extracted from its content. Subsequently, in Section 4.2 we define a new task in the domain of open information extraction called *fact salience*, which addresses the objective of generating a machine-readable summary constituted of *salient facts* extracted from a given document. Furthermore, in Section 4.2.2, we propose SalIE, the first fact salience system that implements a novel unsupervised algorithm for *ranking* and *diversifying* the extracted open facts.

4.0.1 Related Work

In this section we jointly present the related work for both *entity* and *fact salience* tasks. The first part of this section introduces the research area of automatic document summarization, while the two following parts describe works that are related to entity and fact salience problems, respectively.

Automatic Document Summarization is the umbrella topic for both entity and fact salience. Given an input document, summarizers aim at generating a text that is shorter than the original one but still preserving the salient elements that it conveys, such as sentences, keywords, entities and facts. According to Gambhir & Gupta (2017), the research area of document summarization can be clustered among different dimensions:

- *Objective.* The goal of a document summarizer can be the extraction of a generic or query-focused summary. Generic summarizers aim at computing a summary that generally describes the body of an input document (Ouyang et al., 2013), whereas query-focused summarizes aim at creating a summary with respect to a specific set of query terms (Baumel et al., 2014).

- *Atomic Units.* Document summarizers can work at different levels of granularity over the input document. Summaries are actually generated by running their algorithms over a decomposition of the document content. Sentences (Mihalcea & Tarau, 2004), keywords (Hasan & Ng, 2014; Lahiri et al., 2017; Paranjpe, 2009) or named entities (Gamon et al., 2013) are common atomic units used for decomposing the source text.

- *Approach.* Two main approaches are present in literature. Extractive summarizers (Mihalcea & Tarau, 2004) create a summary through a selection of the atomic units identified in the input document, whereas abstractive summarizers (See et al., 2017) can generate novel units not necessary featured in the source text.

- *Documents.* Single document summarization summarizes the content of one input document (Durrett et al., 2016), whereas multi-document summarization aims at generate a summary from a collection of documents (Nayeem & Chali, 2017).

The algorithms that we propose in this chapter are designed in the domain of generic, extractive and single-document summarization, and they differ for the atomic units they use. Wikipedia entities (extracted via entity linking) and open facts (extracted via open information extraction) are the two atomic units that we respectively use for entity and fact salience tasks, respectively.

Entity salience is a recently introduced task (Dunietz & Gillick, 2014) that aims at solving the summarization problem through the extraction of *salient Wikipedia entities* from an input document. The task has been attacked from two research groups with the design of two different systems: CMU-GOOGLE (Dunietz & Gillick, 2014) and SEL (Trani et al., 2018) systems.

The first approach uses a proprietary entity linker to extract entities from the input text and a binary classifier based on very few and simple features to distinguish between salient and non-salient entities. Dunietz & Gillick (2014) have shown that their system significantly outperforms a simple baseline via some experiments executed over the large and well-known New York Times dataset (Sandhaus, 2008).

The second approach, called SEL, called was proposed by Trani et al. (2018) and hinges on a supervised two-step algorithm comprehensively addressing both entity linking and salience detection. The first step is based on a classifier aimed at identifying a set of candidate entities that are mentioned in the input document, thus maximizing the precision without hindering the recall; the second step is based on a regression model that aims at scoring those candidate entities thus identyfing the top (salient) ones. Unfortunately SEL was compared only against *pure* entity linkers (such as TAGME), which are not designed for the entity salience task, the system is yet publicly unavaiable and, furthermore, its experimental figures were confined to a new dataset (i.e., Wikinews), which was much smaller than NYT, and thus missed a comparison against the CMU-GOOGLE system.

As a result, the two entity salience systems above are not publicly available and their experimental figures are incomparable. In this chapter we continue the study of the entity salience problem by introducing a novel system, that we call SWAT, whose main goal is to efficiently and efficaciously address these open issues by improving the state-of-the-art.

Fact salience is a new task that we propose in this chapter and that aims at solving the summarization problem through the extraction of *salient open facts* from an input document. Open facts have been already used in document summarization for avoiding redundancy, exploiting synonymity (Christensen et al., 2013) or as input for a classifier (Christensen et al., 2014), but not used as atomic units as we do in our context. As we will show, working at the fact level provides a natural framework for distilling essential information from an input document. Since facts are minimal comprehensive atomic units expressing a single proposition (Del Corro & Gemulla, 2013), they naturally help to avoid working with sentences that might express more than one proposition or arbitrary chunking of the input document. Compression is also more principled at a fact level as the fact structure is clearly defined (Gashteovski et al., 2017). Additionally, we exploit the fact structure to promote diversity.

The approaches proposed by Mihalcea & Tarau (2004) and Erkan & Radev (2004) are similar to the framework that we devise for solving the fact salience problem. Indeed, we use PageRank for establishing the relative prominence of facts but, unlike from Mihalcea & Tarau (2004) and Erkan & Radev (2004), we weight graph edges using word embeddings to allow more expressive semantics and thus avoiding the sparsity frequency-based methods, as well as we personalize the teleport vector of PageRank with a prior relevance scoring function. Different methods have also been explored to promote diversity: for example, Xiong & Luo (2014) uses LSA and Chien & Chang (2013) relies on topic models, whereas, in our case, we generate diversity by exploiting the fact structure.

Fact salience is also related to triple scoring in KGs (Bast et al., 2017). However, in fact salience a fact is not relevant per se but locally in a textual context, whereas triple KG scoring asses the global relevance of a KG fact for a specific entity — e.g. *("T. Burton", "profession", "actor")* vs. *("T. Burton", "profession", "director")*.

Summarizing, the research contribution of this work is twofold. First, we introduce *fact salience*: the task of generating a machine-readable representation of the most prominent information in an input document as a set of facts. Second, we propose SalIE, the first fact salience system known in the scientific literature. SalIE is compared with several baselines (including positional, standard for salience tasks) with state-of-the-art automatic text summarizers. SalIE outperforms baselines and text summarizers showing that facts are an effective way to compress information.

4.1 Entity Salience

In this section we propose our solution for the entity salience task. More precisely, we design a novel entity salience system called Swat (**S**alient **W**ikipedia **A**nnotation of **T**ext), which constitutes the state-of-the-art in extracting salient Wikipedia entities occurring in an input text. The software architecture of Swat relies on a pipeline organized in three main modules: *Document Enrichment*, *Feature Generation* and *Entity Salience Classification*.

Given an input document, the *Document Enrichment* module enriches the source text with proper syntactic, semantic and latent elements that are automatically extracted through the deployment of four software components: (1) CoreNLP (Manning et al., 2014), the most well-known NLP framework to analyze the grammatical structure of sentences, is used to extract the morphological information coming from the dependency trees built over the sentences of the input document; (2) Wat (Piccinno & Ferragina, 2014), one of the best publicly available entity linkers (Usbeck et al., 2015), is used to extract Wikipedia entities from an input text and to build an entity graph for weighting the importance of these entities and their semantic relationships; (3) TextRank (Mihalcea & Tarau, 2004), the popular document summarizer, is used to return a score for each sentence of the input document; and (4) Word2Vec, the continuous vector space representation of words and entities captured via deep neural networks, is used to enrich the entity graph of point (2) with distributional latent signals. Subsequently, the *Feature Generation* module dispatches the element extracted from the first stage to a number of other software components in order to map each entity into its proper vector of features, which significantly expands the ones investigated in previous papers (Dunietz & Gillick, 2014; Trani et al., 2018). Finally, these feature vectors are given as input to the *Entity Salience Classification* module that leads to discriminate entities into salient and non-salient.

Our system is evaluated through a large experimental assessment executed over two datasets, known as New York Times and Wikinews. Swat will be compared against two systems that constitute the state-of-the-art in this setting, namely Cmu-Google (Dunietz & Gillick, 2014) and Sel (Trani et al., 2018). This experimental study will show that Swat raises the best known performance in terms of F1 up to 3.4% (absolute) over Cmu-Google system and up to 6.3% (absolute) over Sel system the two experimented datasets. These F1-results will be complemented with a throughout discussion about the impact of each feature onto the overall performance of our system and on how the position of salient entities does influence the efficacy of their detection. In this latter setting, we will show that the improvement of Swat with respect to Cmu-Google over the largest dataset New York Times may get up to 14% in micro-F1.

In summary, the main contributions of this Section are the following ones:

- We design and implement Swat, an effective entity salience system that detects the salient entities of a document via novel algorithms that extract a rich set of features: syntactic (sentences' ranking, dependency trees, etc.), latent (i.e., word and entity embeddings), and semantic (computed via a new graph representation of entities and several centrality measures). Despite the use of word and entity embeddings is not new in NLP and IR domains, we are the first (to the best of our knowledge) to investigate its effectiveness on the entity salience task with a proper engineering of features based on these latent representations of entities.

- We are also the first ones to offer an extensive experimental comparison among all known entity salience systems (i.e., Swat, Sel and Cmu-Google, plus several other baselines) over the available datasets: i.e., New York Times (Dunietz & Gillick, 2014) and Wikinews (Trani et al., 2018).

- These experiments show that Swat consistently improves the F1 performance of Cmu-Google and Sel over those two datasets by achieving, respectively, an improvement of about +12.2% (absolute) and +6.3% (absolute).
- These figures are accompanied by a thoughtful analysis of Swat's features, efficiency and errors, thus showing that all of its components are crucial to achieve its improved performance both in F1 and time efficiency.
- In order to encourage the development of other research built upon entity salience tools, we release Swat as a public API[*], which actually implements the full entity salience extraction pipeline thus ease its plugging into other software.

4.1.1 Our Proposal: Swat

In this section we describe our system Swat, which aims at extracting the salient Wikipedia entities of an input document through a pipeline of three main modules: *Document Enrichment*, *Feature Generation* and *Entity Salience Classification*. A graphical representation of Swat is provided by Figure 4.1.

Document Enrichment. The first module aims at extracting from input document d a set of syntactic, semantic, and latent information. Specifically, this module is organized in four main components:

1. CoreNLP (Manning et al., 2014) is the component in charge of enriching the document with proper morphological NLP extractions. Specifically, it tokenizes the input document d, assigns a part-of-speech-tag to each token, generates the dependency relations between the tokens, identifies proper nouns and finally generates the coreference chains.
2. TextRank (Mihalcea & Tarau, 2004) is a component that works by taking as input the sentences tokenized by CoreNLP and by ranking them via a random walk over a complete graph in which nodes are sentences and the weights of the edges are computed as a function of the normalized number of common tokens between the connected sentences.

[*] sobigdata.d4science.org/web/tagme/swat-api

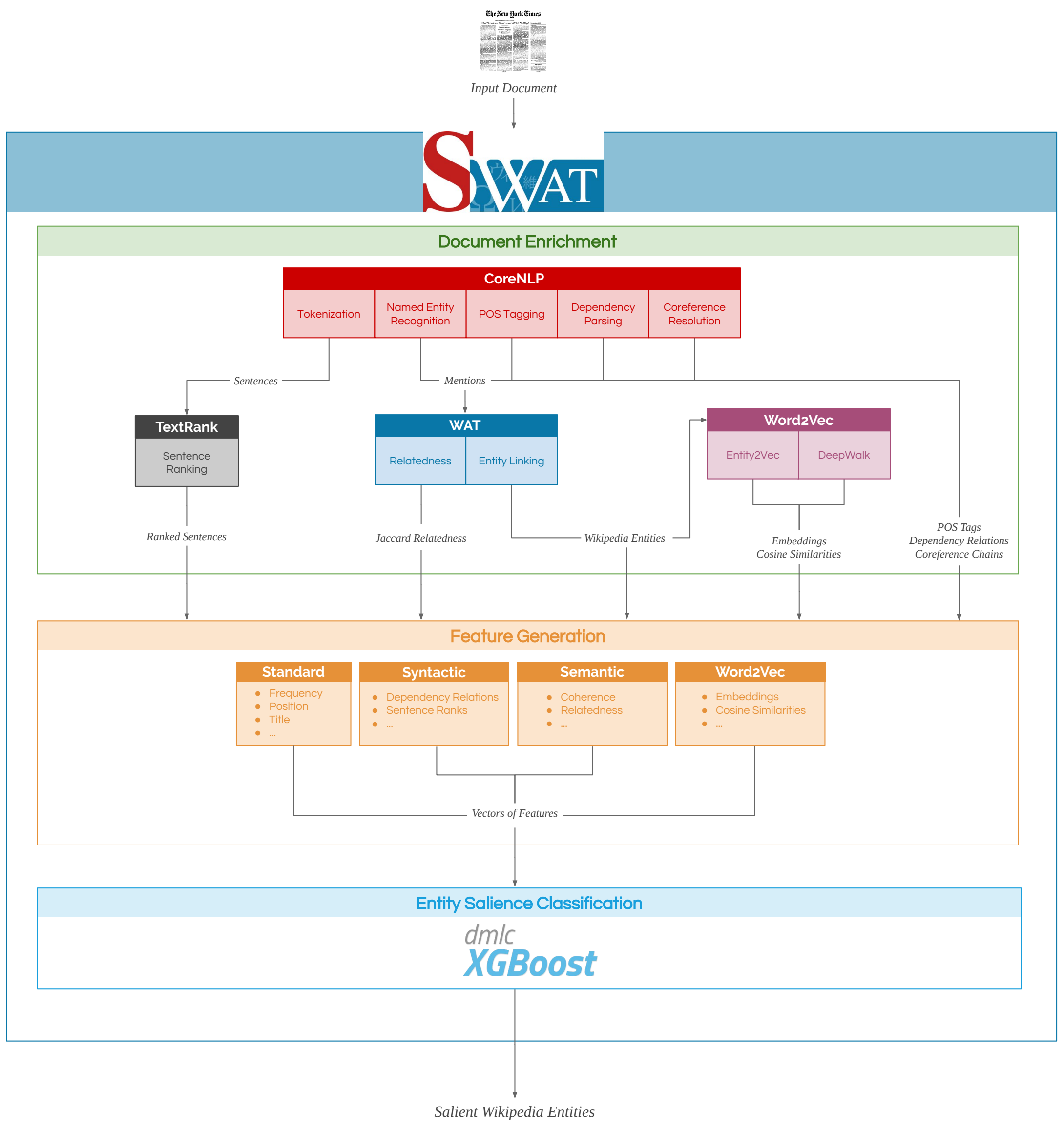


Figure 4.1: Three-module architecture of SWAT that implements the entity salience pipeline.

3. WAT (Piccinno & Ferragina, 2014) is the component that aims to annotating *d* with a set of Wikipedia KG-based extractions (m, e), where m is a sequence of words (i.e., *mentions*, provided by CoreNLP as proper or common nouns) and e is an entity (i.e., Wikipedia page). Specifically, WAT disambiguates every mention m by assigning to each mention an entity provided with two main scores:

(a) *commonness*, which represents the probability that m is disambiguated by e;

(b) *coherence* (denoted by ρ), which represents the semantic coherence between the extraction and its textual context in d.

Subsequently, this component generates an *entity graph* in which nodes are the extracted entities and edges are weighted with the Jaccard relatedness (see Chapter 3) between the edge-connected entities.

4. Word2Vec is the component that aims to enrich the document with latent information. More precisely, it takes the entities extracted by WAT and map them into their proper continuous vector representations learned via neural networks (Mikolov et al., 2013b). These latent representations are further used to compute the cosine similarities between all entities that have been extracted in the document d by WAT. Technically speaking, the Word2Vec component is constituted by two sub-components that respectively deploy two different kinds of *latent entity representations*: Entity2Vec (Ni et al., 2016) and DeepWalk (Perozzi et al., 2014), respectively.

Feature Generation. The second module deploys the data generated by Document Enrichment in order to compute a rich set of features for each entity e. Four main components are here deployed (i.e., Standard, Syntactic, Semantic and Word2Vec in Figure 4.1) in order to map each e into its proper vector of features. A more detailed description of these components as well as the algorithms implemented to generate the features for each entity is provided below.

Entity Salience Classification. The goal of the last module is to classify entities into their class (i.e., salient vs non-salient) given the entity features computed by the previous module. To accomplish this task, we deploy the gradient tree boosting implementation provided by the efficient and highly scalable *eXtreme Gradient Boosting* software library (Chen & Guestrin, 2016) (XGBOOST in Figure 4.1) which is trained and tested as detailed in Section 4.1.2.

More on Feature Generation

Although the use of the third module is quite standard, the first and second modules are more involved and constitute the main novel part of our system SWAT. Hence, the rest of

Table 4.1: Standard features adopted by SWAT.

Name	**Description**	**Component**
$ef(e, d)$, $idf(e)$, $ef\text{-}idf(e, d)$	Entity frequency (number of times WAT extract e in d), inverse document frequency for e and their product.	Standard
$position\text{-}stats_{\{s,t\}}(e, d)$	Minimum, maximum, arithmetic mean, median, standard deviation and harmonic mean of sentence- (resp. token-) positions of e in d.	Standard
$mention\text{-}title(e, d)$	Presence of a *mention* of e in the title of d.	Standard
$entity\text{-}title(e, d)$	Presence of e in the title of d.	Standard
$is\text{-}upper(e, d)$	True if one of the mentions of e appear in d in uppercase, false otherwise.	Standard

Table 4.2: Features introduced by the CMU-GOOGLE system (Dunietz & Gillick, 2014) and adopted by our system SWAT.

Name	**Description**	**Component**
$1st\text{-}loc(e, d)$	Index of the sentence in which the first mention of e appears in d.	Standard
$head\text{-}count(e, d)$	Frequency of head word of entity e in the document d.	Syntactic
$mentions(e, d)$	Sum between entity frequency and co-referenced frequency of e in d.	Syntactic
$headline(e, d)$	POS tag of each word of e that appears in at least one mention and also in the headline of d.	Syntactic
$head\text{-}lex(e, d)$	Lower-cased head word of the first mention of e in d.	Syntactic
$google\text{-}centrality(e, d)$	PageRank score of e on the entity graph generated from d, where weights are the co-occurrence probability of two entities, computed on the training set.	Standard

this section is devoted to detail the first two modules which generate the features for each entity that has been annotated in the input document d — called Standard, Syntactic, Semantic and Word2Vec — to be used in the third and last entity salience classification module. In order to facilitate the reading and understanding of the large number of features deployed by SWAT, we report all of them in Tables 4.1, 4.2 and 4.3 which respectively group those features by novelty and by the software component which is in charge of their implementation (rightmost column in each table). In the text below we comment only on the new features

Table 4.3: Novel features introduced by SWAT.

Name	Description	Component
$spread_{\{s,t\}}(e,d)$	Difference between the max and min sentence- (resp. token-) positions of *e* in *d*.	Standard
$bucketed\text{-}freq_{\{s,t\}}(e,d)$	Vector of bucketed frequencies through sentence- (resp. token-) positions of *e* in *d*.	Standard
$textrank\text{-}stats(e,d)$	Minimum, maximum, arithmetic mean, median, standard deviation and harmonic mean of TextRank scores of sentences where *e* appears in *d*.	Syntactic
$dep\text{-}freq(e,d)$	Frequency of *e* in *d* when it appears as dependent of the dependency relation *dep*.	Syntactic
$dep\text{-}bucketed\text{-}freq_{\{s,t\}}(e,d)$	Vector of bucketed frequencies through sentence- (resp. token-) positions of *e* in *d* limited to the mentions where *e* appears as dependent with relation *dep*.	Syntactic
$dep\text{-}position\text{-}stats_{\{s,t\}}(e,d)$	Minimum, maximum, arithmetic mean, median, standard deviation and harmonic mean of sentence- (resp. token-) positions of *e* in *d*, where only the mentions where *e* appears as dependent of a dependency relation *dep* are considered.	Syntactic
$dep\text{-}textrank\text{-}stats(e,d)$	Minimum, maximum, arithmetic mean, median, standard deviation and harmonic mean of TextRank scores where only the sentences where *e* appears as dependent of a dependency relation *dep* are taken into account.	Syntactic
$comm\text{-}stats(e,d)$	Minimum, maximum, arithmetic mean, median, standard deviation and harmonic mean of the *commonness* values of *e* in *d* computed by WAT.	Semantic
$\rho\text{-}stats(e,d)$	Minimum, maximum, arithmetic mean, median, standard deviation and harmonic mean of the ρ-score values of *e* in *d* computed by WAT.	Semantic
$rel\text{-}stats(e,d)$	Minimum, maximum, arithmetic mean, median, standard deviation and harmonic mean of the relatedness scores between *e* and all other entities extracted in *d*.	Semantic
$rel\text{-}bucketed\text{-}stats_{\{s,t\}}(e,d)$	Minimum, maximum, arithmetic mean, median, standard deviation and harmonic mean of the relatedness scores between *e* and all other entities present in *d*, bucketed over document positions (both at sentence- and token-level).	Semantic
$rel\text{-}centrality(e,d)$	Degree, PageRank, betweenness, Katz, HITS, closeness and harmonic scores of *e* computed on the entity graph of *d*.	Semantic
$wiki\text{-}id(e)$	Wikipedia identifier of *e*, normalized via feature hashing.	Semantic
$w2v(e)$	Entity2Vec and DeepWalk embedding vectors of *e*.	Word2Vec
$w2v\text{-}stats(e,d)$	Minimum, maximum, arithmetic mean, median, standard deviation and harmonic mean of the cosine similarity between the Entity2Vec and DeepWalk embeddings of *e* and the ones of the other entities extracted in the title and headline of *d*.	Word2Vec
$w2v\text{-}cos\text{-}title(e,d)$	Cosine similarity between the Entity2Vec and DeepWalk embeddings of *e* and the average of the corresponding embeddings of the words present in the title of *d*.	Word2Vec
$w2v\text{-}cos\text{-}headline(e,d)$	Cosine similarity between the Entity2Vec and DeepWalk embeddings of *e* and the average of the corresponding embeddings of the words present in the headline of *d*.	Word2Vec

introduced by Swat, referring for the others to the description reported in the tables. And, for each novel set of features, we first report their technical description and then introduce a specific paragraph in which we will detail the motivations and the phenomena that the new engineered features will aim to capture.

Position-based Features. These features deploy the distribution within document d of the entities occurrences in order to predict their salience score. Furthermore, all position features of an entity e within the document d are computed by Swat in terms of *tokens* or *sentences*. For token-level features (indicated with the subscript t) it is considered the index of the first token for each mention of e, normalized by the number of tokens of d; whereas for sentence-level features (indicated with the subscript s) it is considered the index of the sentences where the entity e is annotated, normalized by the number of sentences of d. These features naturally improve the $1st\text{-}loc(e,d)$ feature introduced by Dunietz & Gillick (2014) thus making more robust Swat with respect to the distribution of salient entities, as shown in Section 4.1.2.

Captured Phenomena. Through the proposal of this set of features (expressed both at sentence- and token-level) we aim at capturing finer-grain positions of the annotated entities than the one previously proposed by (Dunietz & Gillick, 2014) with $1st\text{-}loc(e,d)$. More precisely, $1st\text{-}loc(e,d)$ is calculated by modeling the sentence index i as an array of size 10 where all elements are 0 except the one at index $log(10 \cdot (i+1))$ (see (Dunietz & Gillick, 2014) for details). This normalization technique clearly has several disadvantages: (1) it lacks in distinguishing the position among entities annotated within the same sentence, (2) it assigns the same value to entities that are annotated in different sentences, and (3) it totally ignores the full length of the text. A graphical example that shows these limitations and how token- and sentence-level features can solve this problem is shown in Figure 4.2. As we can see from that figure, the two entities `Barack_Obama` and `Hilary_Clinton` appear in the same sentence but at different positions, so that $1st\text{-}loc$ and $position\text{-}min_s$ get the same values but $position\text{-}min_t$ allows to differentiate them. Furthermore, `Iraq` and `New_Hampshire`, which have been respectively annotated in the 4th and 7th sentences, achieve the same $1st\text{-}loc$ values but $position\text{-}min_s$ and $position\text{-}min_t$ get different values.

One more issue that afflicts the $1st\text{-}loc$ feature consists in the fact that it models only the *first* position of an entity e in d, thus failing in capturing the (possibly meaningful) distri-

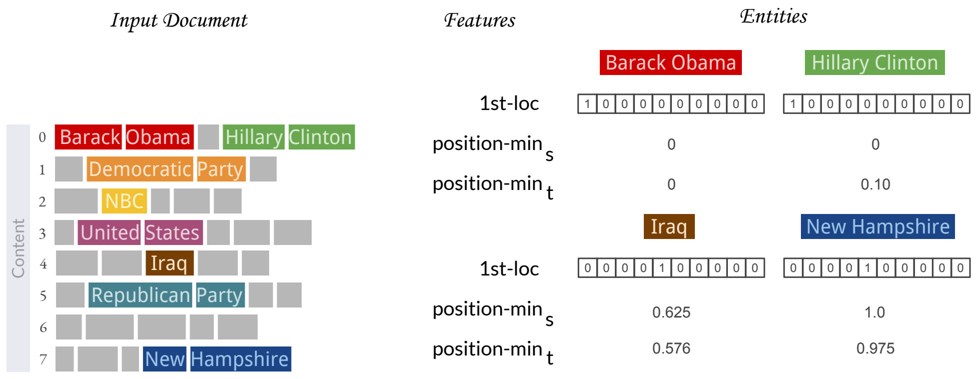


Figure 4.2: Example that shows where *1st-loc* feature fails in a proper differentiation among entities' positions. Numbers on the left of the document are the sentence indices, rectangles represent tokens in the input document and entities are tinted with different colors (see comment in the text).

bution of that entity in the input document. Figure 4.3 and 4.4 show the distribution of salient versus non-salient entities among two different datasets. As we can see, salient entities present a common pattern, with a frequency that is very high at the beginning and smoothly decreases among the rest of the document. Accordingly, we decided to investigate the computation of two specific features that should model this phenomenon: *bucketed-freq* captures the distribution of an entity in the input text, and *spread* computes the difference between the position of the first and the position of the last mention of an entity in the input text.

Summarization-based Features. These features exploit the score that summarization algorithms assign to sentences that contain salient information and thus possibly contain salient entities. Accordingly, SWAT computes, for each entity *e*, several statistical measures derived from the scores assigned by TextRank (Mihalcea & Tarau, 2004) to the sentences where a mention of *e* occurs.

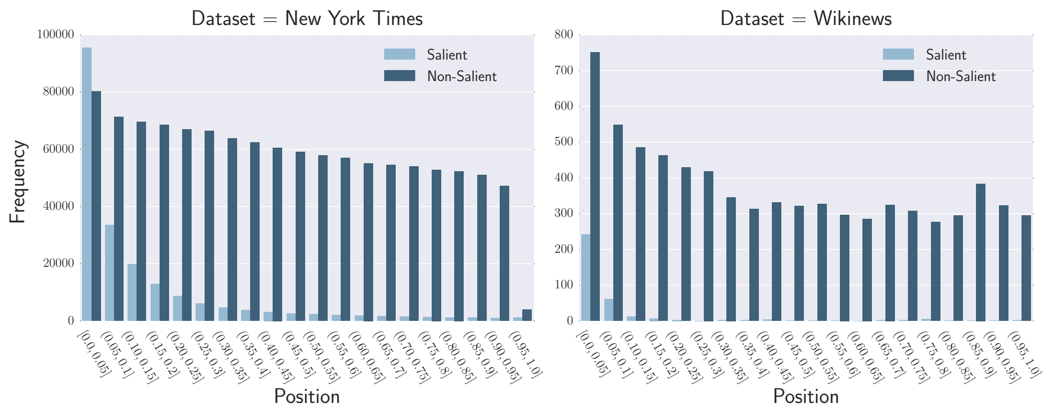


Figure 4.3: The histograms plot the frequency distribution of salient versus non-salient entities according to their *first* positions in the documents over NYT (left) and Wikinews datasets (right).

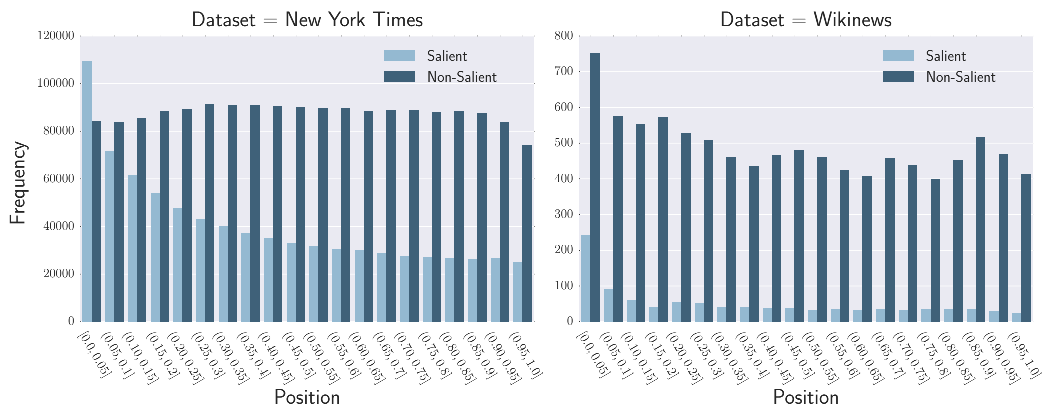


Figure 4.4: The histograms plot the frequency distribution of salient versus non-salient entities according to *all* their occurrences in the documents over NYT (left) and Wikinews datasets (right).

Captured Phenomena. These features aim at capturing the syntactical centrality of a sentence in a document and postulate that *"salient entities are contained in sentences which are central for the input document"*. This "centrality issue" is a signal commonly used by popular state-of-the-art text summarizers (Mihalcea & Tarau, 2004). We implemented this idea by defining a feature that assigns high scores to entities which occur in sentences which are highly rated by TextRank.

Linguistic-based Features. These features exploit dependency trees of the sentences where the entities occur. Unlike Dunietz & Gillick (2014), where dependency trees are used to extract only the head of a mention, SWAT combines frequency, position and summarization information with several dependency relations generated by CoreNLP's dependency parser.

Captured Phenomena. Through these features we aim at modelling the morphological associations (i.e., dependency relations) among the entity's mentions in a text. Specifically, a number of mentions of salient entities in the benchmarked datasets frequently have tokens which are dependent of preposition-in, adjective modifier, possessive, noun compound modifier and subject dependency relations. Accordingly, we designed features that compute position, frequency and sentence scores by filtering only the mention of entities whose tokens appear as dependent of one of the dependency relations mentioned above (i.e., *dep-*$*$ features).

Annotation-based Features. This set of features computes several statistics upon *commonness* and ρ scores which have been assigned to each annotation (m, e) by the entity linker WAT. These two scores actually capture two different aspects of the coherence of (m, e) within the input text where it has been detected: *commonness* provides a sort of common-sense probability that m can be disambiguated with e, whereas ρ quantifies the quality of the annotation in terms of coherence between e and its context of occurrence in the input document d.

Captured Phenomena. Although entity linkers currently reach very good performance on different datasets (see (Usbeck et al., 2015) for details), they can also incur into several errors by annotating a mention m with a misleading entity e. In the entity salience problem, a wrongly annotated entity can introduce some noise in the entity salience pipeline, with a worst-case scenario where the misleading entity is eventually classified as salient. To limit the impact of entities wrongly detected by WAT, we decided to extract several other features based on the *commonness* and ρ scores (resp. *comm-* and *ρ-stats* features) with the intuition that these scores should increase the robustness of our entity salient classifier.

Word2Vec-based Features. This set of features aims at modeling the annotated entities and their relationships by means of proper embeddings generated via deep neural networks. Specifically, SWAT deploys both CBOW and Skip-gram models, generated by Entity2Vec and DeepWalk, here trained on the Wikipedia KG as previously described in Chapter 3.
SWAT uses as features the continuous vectors derived from Entity2Vec and DeepWalk, plus several other statistics computed over their cosine similarity measure.

Captured Phenomena. The features built on top of the Word2Vec component aim at capturing those kinds of latent signals that can not be explicitly detected from the syntactic and

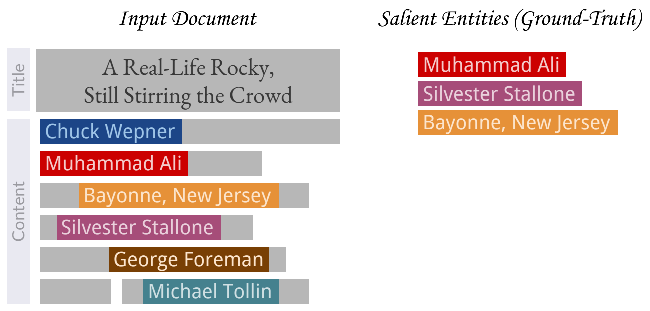


Figure 4.5: Example where the latent information between the title of the input document and the candidate entities can help in distinguishing between salient and non-salient entities. In fact, despite `Silvester_Stallone` and `Muhammed_Ali` do not explicitly appear in the document's title, their relationship with Rocky is a clear strong indication of their salience.

morphological features proposed before. For example, the latent relationships between the title of the input document and the candidate entities should help our system to correctly detect the correct entities, especially when they are salient and do not appear at the beginning. A graphical example is reported in Figure 4.5. As we can see, the title contains some information that can help SWAT into the correct classification of `Silvester_Stallone` and `Muhammed_Ali` as salient.

Relatedness-based Features. These features are introduced to capture how much an entity e is related to all other entities in the input document d, with the intuition that if an entity is salient then its topic should not be *isolated* in d. SWAT uses two main groups of relatedness functions:

1. the Jaccard relatedness described by Piccinno & Ferragina (2014), since its deployment in the disambiguation phase of WAT achieves the highest performance over different datasets (Usbeck et al., 2015);
2. the cosine similarity between the latent embeddings of the compared entities, either based on Entity2Vec or on DeepWalk.

Furthermore, these relatedness functions are used to compute other two classes of features:

1. the ones based on several centrality algorithms – i.e., degree, PageRank, betweenness, Katz, HITS, closeness and harmonic (Boldi & Vigna, 2014) – applied over three versions of the entity graph described in Stage 1. We recall that this is a complete graph

where nodes are entities and edges are weighted with a relatedness measure between the connected entities which is estimated either with Jaccard, Entity2Vec or DeepWalk.

2. the ones based on proper statistics aggregating the relatedness scores between the entity *e* and other entities in *d*.

Captured Phenomena. Through these features we aim at capturing how much an entity is semantically central with respect to the other annotated entities. Figure 4.6 shows an intuitive example where the centrality of entities should play a role in discriminating between salient and non-salient entities. More precisely, highly related entities will receive higher centrality scores (i.e., `New_York_City` and `Fashion_Week`), whereas the ones that are poorly related with the others (i.e., `Lower_East_Side`) will receive lower centrality scores and hence should be classified as less salient for the content of the input document.

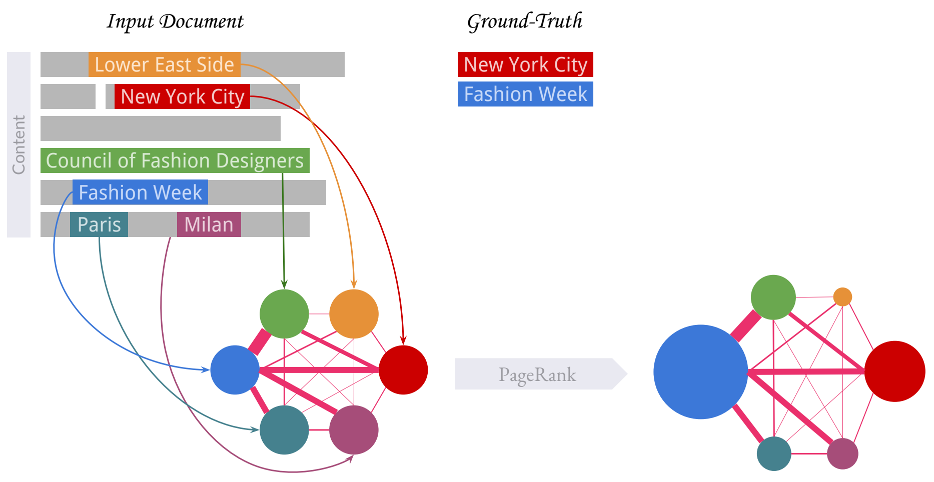


Figure 4.6: A graph where nodes are entities annotated in the input document and edges are weighted with the relatedness score between the connected entities. Ticker edges indicate higher weights. Centrality scores are eventually computed by running PageRank, and they should show how the relatedness-based feature can help in distinguishing salient and non-salient entities.

According to (Boldi & Vigna, 2014), centrality can actually be defined in several ways and literature currently does not offer a uniform terminology as well as different centrality algorithms capture different aspects of nodes and their connections in the graph. Degree offers a "majority voting" between nodes, PageRank computes the probability that a random surfer passes into a node by intermittently teleporting back to other nodes, betweenness measures the volume of the shortest paths passing through a given node, Katz sums the weighted paths coming into a node, HITS scores each node with a high authoritative (resp. hub) value whether the node at hand is pointed by many good hub (resp. authoritative node), closeness assigns a higher score to nodes that have smaller distance with respect to all the others in the graph, and harmonic measures the harmonic mean of all distances between every pair of nodes. Because it is unknown what kind of centrality algorithms could be more effective in the context of entity salience, we decided to investigate the use of all of them over the graph of entities described above.

4.1.2 Experiments

In this section we describe the experimental methodology performed for evaluating our system Swat. First, we introduce the datasets used in our benchmarks by reporting the main differences between the two test-beds. Second, we outline the metrics used in the experiments for measuring the accuracy of the systems at hand and finally we describe and discuss the results of the the experimented systems.

Datasets

The assessment of the accuracy and efficiency performance of Swat is executed on the following datasets.

New York Times. The annotated version of this dataset, suitable for the entity salience problem, was introduced by Dunietz & Gillick (2014). It consists of annotated news drawn from 20 years of the New York Times newspaper — see also (Sandhaus, 2008). It is worth to point out that the numbers reported by Dunietz & Gillick (2014) are slightly different from the ones we derived by downloading this dataset: authors informed us that this is due to the way they have exported annotations in the final release and this impacts onto the F1-performance of their system for about -0.5% in absolute micro-F1. We will take these figures into account in the next sections when comparing Swat with the Cmu-Google system. Since the entity linker used by Dunietz & Gillick (2014) is not publicly available (and this was used to derive the ground truth of the NYT dataset), we kept only those entities which have been generated by Swat and Cmu-Google. The final figures are the following: the news in the training+validation set are 99 348 = 79 462+19 886, and are 9577 in the test set; these news contain a total of 1 276 742 entities in the training+validation set (i.e., 1 021 952 + 254 790) and 19 714 entities in the test set. Overall the dataset contains 108 925 news, with an average number of 975 tokens per news, more than 3 million mentions and 1 396 456 entities, of which 14.7% are labeled as salient.

Wikinews. This dataset was introduced by Trani et al. (2018) and consists of a sample of news published by Wikinews from November 2004 to June 2014 and extracted with Wikipedia entities by the Wikinews community. This dataset is significantly smaller than NYT in all means: number of documents (365 news), their lengths (an average of 297 tokens per document) and number of annotations (a total of 4747 manual extracted entities, of which 10% are labeled as salient). Nevertheless, this dataset has some remarkable features with respect to NYT: the ground-truth generation of the salient entities was obtained via human-assigned scores rather than being derived in a rule-based way, and it includes both proper nouns (as in NYT) and common nouns (unlike NYT) as salient entities. For the cleaning of the dataset we follow Trani et al. (2018) as done in their experimental setup by removing the 61 documents that do not have any salient entity.

As far as the dataset subdivision and evaluation process are concerned, we used the following methodology. For the NYT, we use the same training/testing splitting as defined by Dunietz & Gillick (2014) as detailed above. For Wikinews we deploy the evaluation procedure described by Trani et al. (2018), namely the averaged macro-F1 of a 5-fold cross-validation.

Evaluation Metrics

For the evaluation of the accuracy of the systems we use Precision, Recall and F1, as standard in IR (Manning et al., 2008) for the assessment the quality of classification systems. The metrics are clearly calculated by considering salient and non-salient classes as, respectively, positive and negative classes (see Table 4.4).

In our experiments we report both micro and macro scores of these metrics, since they have been respectively used by Dunietz & Gillick (2014) and Trani et al. (2018) for validating their systems. Micro scores focus their evaluation of the overall quality of the salient

Table 4.4: Brief recap of the classification metrics used in our experiments.

		Prediction	
		Salient	Non-Salient
Real	Salient	*tp*	*fn*
	Non-Salient	*fp*	*tn*

Measure	**Equation**
Precision	$\frac{tp}{tp+fp}$
Recall	$\frac{tp}{tp+fn}$
F1	$2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$

predictions in a dataset, whereas macro scores are computed as the average of micro metrics calculated for every single document.

Systems and Baselines

Baselines. We implemented four baselines. The first one is the same baseline introduced by Dunietz & Gillick (2014), it simply classifies an entity as salient if it appears in the first sentence of the input document. The other three baselines are new and try to investigate the individual power of some novel features adopted by Swat. More precisely, the second baseline (called ρ-baseline) extends the previous one by adding the check whether the ρ-score (capturing entity coherence) is greater than a fixed threshold. The third (resp. fourth) baseline classifies an entity as salient if its maximum TextRank (resp. Rel-PageRank) score is greater than a fixed threshold.

Two Versions of the Cmu-Google System. The original system (Dunietz & Gillick, 2014) uses a proprietary entity linker to link proper nouns to Freebase entities, and then classifies them into salient and non-salient by deploying a small number of standard text-based features, mainly based on position and frequency. This system is not available to the public, so we will report in our tables the performance figures published by Dunietz & Gillick (2014). To support experiments over the new dataset Wikinews, we decided to implement our own version of the Cmu-Google's system by substituting the proprietary modules with open-source tools: we used Wat as entity linker (Piccinno & Ferragina, 2014) and a state-of-the-art logistic regressor as classifier (Pedregosa et al., 2011). Our (re-)implementation achieves performance very close to the original system (see Table 4.6) and thus it is useful to obtain a fair comparison over the Wikinews dataset.

Table 4.5: Candidate values and the best configuration found by the grid-search procedure for the tuning of XGBoost's hyper-parameters on New York Times and Wikinews datasets.

Hyper-parameters	Candidate Values	New York Times	Wikinews
max_depth	$\{2, 4, 6, 8\}$	8	2
min_child_weight	$\{6, 8, 10\}$	6	6
gamma	$\{0.1, 0.3, 0.5\}$	0.1	0.5
reg_alpha	$\{0.001, 0.01, 0.05\}$	0.001	0.05
scale_pos_weight	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$	2	8

The Sel System. This is the system proposed by Trani et al. (2018) that uses a machine learning regressor to detect salient entities via a set of features that is wider than the ones used in Cmu-Google. This system is not available to the public, so we will report in our tables the performance figures published by Trani et al. (2018).

Configurations of Swat and Baselines. We experimented upon several configuration settings of Swat and of the baselines above, according with the characteristics of the ground-truth datasets. For NYT, where the ground-truth was generated by assuming that salient entities can be mentioned in the text only as proper nouns, we configured these systems to extract only proper nouns detected by CoreNLP; whereas for Wikinews, where the ground truth comes with no assumptions, we tested two variants: one detecting only proper nouns, and the other detecting both proper and common nouns. For the tuning of XGBoost's classifier we performed a grid-search over typical values of its hyper-parameters, finding the *best values* (i.e., the ones performing better on the validation sets of New York Times and Wikinews) reported in Table 4.5.

Results and Analysis

We first experiment our proposed Swat against the state-of-the-art tools over the two datasets New York Times and Wikinews, and then we complement these results with a supplementary analysis and discussion of several aspects of our proposed system. Specifically, we will focus on: (1) the generalization ability of the tested systems as a function of the used training data, (2) the dependence between the size of the training-set and the accuracy of the system, (3) the impact that features have on the quality of the predictions, (4) the time efficiency of the system according to its main components and its overall speed-up when only the most relevant features are used, (5) the dependence of top-systems on the position of the salient entities within the input document, and (6) an analysis of the limitations of the current systems in terms of the types of erroneous predictions.

Results. Experimental figures on the two datasets are reported in Tables 4.6–4.7, where we denote by Cmu-Google-ours our implementation of the system by Dunietz & Gillick (2014). This system is only slightly worse than the original one, which could depend on the differences in the NYT dataset commented above and in the deployment of open-source modules rather the Google's proprietary ones. The final performance of Cmu-Google-ours is very close to what claimed by Dunietz & Gillick (2014), thus we decide to use this software also on the Wikinews dataset. We notice that both TextRank and Rel-PageRank baselines obtain low micro- and macro-F1 performance over both datasets. This is probably due to the characteristics of these datasets: the salient information in news is typically confined to initial positions, so those systems are drastically penalized by ignoring positional information. This statement is further supported by the results of Positional and Positional-p baselines: they are trivial but generally achieve better performance.

Table 4.6 reports the results for the experiments on the New York Times dataset. We notice that the new features adopted by Swat allow it to outperform Cmu-Google-ours by 3.4% and 3.3% over micro- and macro-F1, respectively, and Cmu-Google by 2.6% in micro-F1 — macro-F1 was not evaluated by Dunietz & Gillick (2014). We tested statistical significance with respect to Cmu-Google-ours[*] using a two-tailed paired t-test and we found that all the improvements reported by Swat in Table 4.6 are statistically significant with $p < 0.01$.

Table 4.7 reports the results on Wikinews dataset. It goes out without saying that the improvement achieved by Swat against the state-of-the-art is even more large than on NYT. Specifically, Swat improves the micro-F1 of 12.2% with respect to Cmu-Google-ours and the macro-F1 of 6.3% with respect to Sel.

[*]Since the original Cmu-Google system is not available we can not test statistical significance with respect to it.

Table 4.6: Performance of the tested systems on the New York Times' dataset. Statistically significant improvements are marked with ▲ for $p < 0.01$.

System	**Micro**			**Macro**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Positional Baseline	59.1	38.6	46.7	39.0	32.7	33.0
Positional-p Baseline	61.9	36.9	46.2	38.5	31.0	32.0
TextRank	27.0	58.8	37.0	30.0	48.6	33.4
Rel-PageRank	20.3	62.5	30.6	21.3	**55.3**	28.0
Cmu-Google	60.5	63.5	62.0	–	–	–
Cmu-Google-ours	58.8	62.6	60.7	47.6	50.5	46.1
Swat	**62.4**▲	**66.0**▲	**64.1**▲	**50.7**▲	53.6	**49.4**▲

Table 4.7: Performance on the Wikinews dataset. For each system we report the score obtained by the system configured to extract either only proper nouns (top) or both proper and common nouns (down).

System	**Micro**			**Macro**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Positional Baseline	23.3	67.0	35.0	25.2	67.0	34.0
	14.4	**72.0**	24.0	16.1	**72.7**	25.0
Positional-p Baseline	36.8	60.3	45.7	38.3	61.6	43.5
	34.1	58.5	43.1	36.2	61.3	41.9
TextRank	12.2	47.5	19.4	14.1	49.3	20.2
	5.7	49.2	10.1	6.3	50.9	10.6
Rel-PageRank	10.0	51.0	16.8	10.1	51.2	15.9
	10.6	35.8	16.4	11.1	34.8	14.7
Cmu-Google-ours	41.0	60.0	49.0	42.3	61.0	46.0
	41.0	56.0	47.0	41.0	58.0	45.0
Sel	–	–	–	**61.0**	50.0	52.0
Swat	**58.0**	64.9	**61.2**	57.7	67.0	**58.3**
	51.0	67.4	58.0	53.7	69.7	56.6

Table 4.8: Generalization ability of Swat trained on NYT and tested on Wikinews. For each system we report the score obtained by the system configured to extract either only proper nouns (top) or both proper and common nouns (down).

System	Micro			Macro		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Swat-clf	35.0	72.0	47.1	37.9	73.7	46.7
	27.3	**75.7**	40.1	31.3	78.0	41.5
Swat-reg	**55.9**	59.9	**57.7**	**54.0**	**62.4**	**54.3**
	49.3	63.1	55.1	50.6	65.9	53.3

Generalization Ability of Swat Trained on NYT. The second question we experimentally investigated is about the *generalization ability* of the feature set used by Swat varying the datasets over which the training and tuning phases are executed. In particular, we experimented two different configurations of our system. Swat-clf is Swat trained over NYT and directly used over Wikinews; and Swat-reg is Swat trained over NYT but whose regressor is tuned over Wikinews maximizing the macro-F1 over the training folds.

According to Table 4.8, Swat-clf obtains performance lower than the systems specifically trained over Wikinews (as expected, see Table 4.7), such as Swat and Sel, but it turns actually to be slightly better than Cmu-Google-ours by +0.7% in macro-F1.

On the other hand, the tuning on Wikinews by Swat-reg allows achieving better performance in macro-F1 than both Cmu-Google-ours and Sel: +8.7% in micro-F1 with respect to Cmu-Google-ours and of +8.3% and +2.3% in macro-F1 with respect to Cmu-Google-ours and Sel. These figures show that the features introduced by Swat are flexible enough to work independently from the news source and without overfitting the large single-source training data (i.e., NYT).

Accuracy versus Training Size. We analyze the performance of the two versions of Swat with respect to different sizes of the training data. We focus these experiments on the largest dataset available, namely New York Times. Figure 4.7 provides a side-by-side comparison of the performance of the two systems when 5%, 25%, 50%, 75% and 100% of the whole training data is used. The original validation set is kept for the tuning of the hyper-parameters.

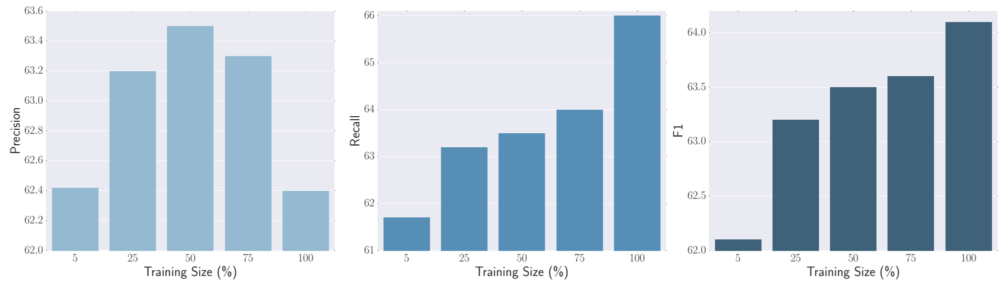


Figure 4.7: Comparison of the performance of SWAT over different training sizes of the New York Times dataset.

Micro-precision, -recall and -F1 are finally calculated over the test-set. The precision of SWAT increases until when 50% of the whole training size is used where it also reaches a peak of 63.5%. Unfortunately, when more than 50% of the training data is used, the precision decreases by eventually losing -1.1% in performance. This degradation is clearly due to the increase of the recall that eventually allows SWAT to consistently improve its micro-F1 until the whole training set is used.

Feature Analysis. Let us jointly discuss the most important signals emerging from the incremental feature additions experimented with SWAT on both datasets (see Figure 4.8). Through this analysis we aim to clarify what are the core elements that are needed for the entity salience detection.

We notice that the most important features for our system depend on four common elements: (1) position (e.g., *position-min*$_t$), (2) the latent similarity between an entity and the title (e.g., *e2v-sg-cos-title*), (3) the centrality of an entity (e.g., *dw- cbow-pagerank* and *dw-cbow-hub*) and finally the (4) coherence scores of the annotated entities (e.g., *comm-max* and *ρ-mean*). On the other hand, frequency signals are fundamental when the input text is large, such as in the NYT dataset (e.g., *head-count* or *mentions*), whereas on relatively shorter documents, such as in Wikinews, they are less useful and they bring improvements only when combined with other signals, such as dependency and positional information (e.g., *sbj-bucketed-freq*$_s$).

We mention here that during this analysis we found several novel errors that are committed by SWAT although its results are better than CMU-GOOGLE system. More precisely it

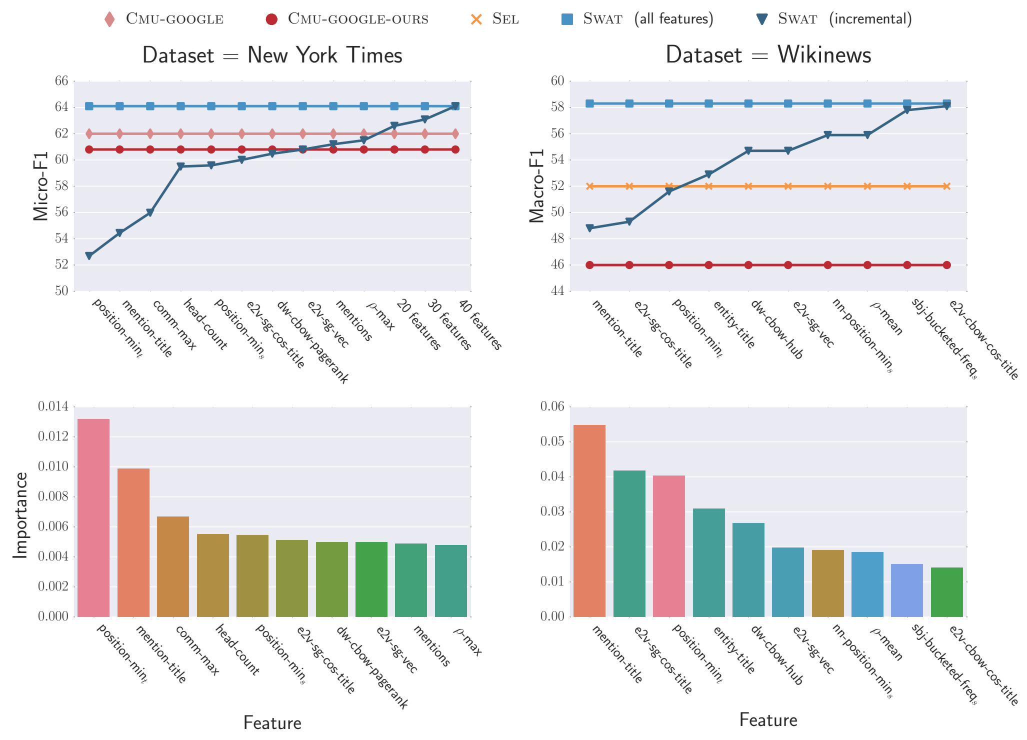


Figure 4.8: Performance of the incremental feature addition (top) of SWAT according to the corresponding feature importance provided by XGBOOST (bottom) over NYT (left) and Wikinews (right) datasets.

is very common that an entity that is salient is present at the beginning of the document, whereas if it appears to fare it is a common practice to classify it as non-salient. On the other hand, we actually found some cases where the features that we designed overcome these problems through our wide feature space that now incorporates more signals than CMU-GOOGLE. Accordingly, we report here several examples where it is evident at human inspection that the designed features help the system in improving its predictions thus showing where our system's predictions mainly differs from CMU-GOOGLE.

For the ease of explanation, we report the whole ground-truth and predictions for both systems, while since input documents are very large we report only several but meaningful annotated entities.

More precisely, we now aim at shadding some light into *how* the new feature space that we designed for SWAT allows our system to achieve higher-quality predictions than CMU-GOOGLE's ones. In accordance with the best features identified by XGBOOST, we report here several practical examples of frequent patterns that we have identified during our analysis and that explicitly show where our *new* and *most relevant* features help SWAT in achieving higher-quality predictions than CMU-GOOGLE. In this analysis we did not consider frequency-based features (i.e., *head-count* and *mentions*) since they are equivalent to the ones already proposed and used by the CMU-GOOGLE system. For ease the understanding of these common patterns, we structured our graphical examples (in Figures 4.9, 4.10, 4.11, 4.12) as follows. On the left, we report a meaningful subset of entities annotated in the input text (since position is a very strong feature, we preserve the order of the annotated entities), in the center we distinguish the two systems (SWAT and CMU-GOOGLE, respectively) and, finally, on the right we report the whole set of predicted salient entities as well as the ground-truth.

Position-based Features. As expected, the new features designed with token-level granularity (i.e., *position-min*$_t$) allow SWAT to achieve a better quality in the detection of salient entities. More precisely, when an entity is mentioned at the beginning of the document (but not in the first few sentences) it is commonly classified by CMU-GOOGLE as non-salient since it obtains a large value for 1*st-loc*. Figure 4.9 shows an example where a salient entity is mentioned at the beginning but, since it appears for the first time only in the third sentence, it is classified by the CMU-GOOGLE system as non-salient. This situation actually repeats frequently in the experimental datasets.

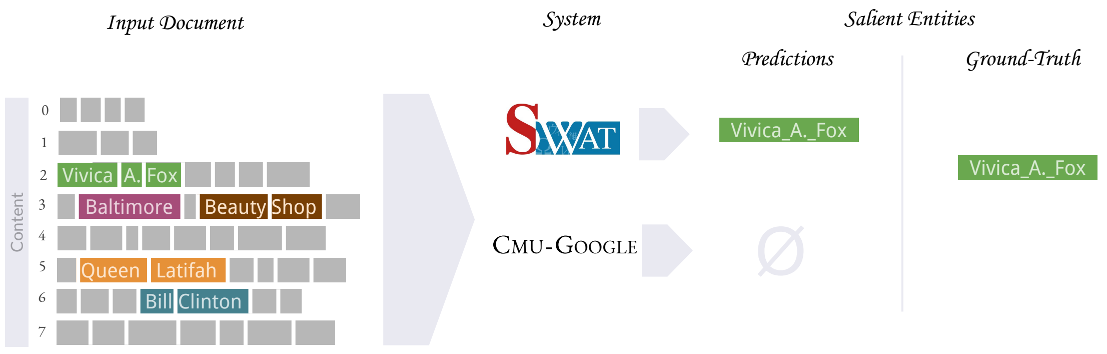


Figure 4.9: Example that shows where token-level features allows SWAT to detect the position of a salient entities as at the beginning of a document although it does not appear in the very first sentences.

On the other hand, the use of a token-level feature allows SWAT to annotate `Vivica_A._Fox` at the very beginning and properly classify it as salient. From our analysis we found that working at token-level makes our system more robust than CMU-GOOGLE, which actually works at sentence-level. Token-level features are more flexible, especially in the cases where the document has several small sentences at the beginning which induce *1st-loc* easily to get large values, as opposite to *position-min*$_t$, which keeps its score low also in these cases.

Title-based Features. In our system we introduced two title-based features (i.e., *mention-title* and *e2v-sg-cos-title*) which aim at improving the quality of the entity-salient classification with information coming from the title of a document. The first feature (i.e., *mention-title*) is actually very simple: when an entity is mentioned in the title it is clearly a strong indication of its salience in the document since the author of the news was probably trying to attract the attention of the reader at first glance. On the other hand, the title can contain information that is related with some entities but without explicitly mentioning them; nevertheless SWAT is still able to capture these related entities and classify them as salient for the input document. An example of this last case is reported in Figure 4.10.

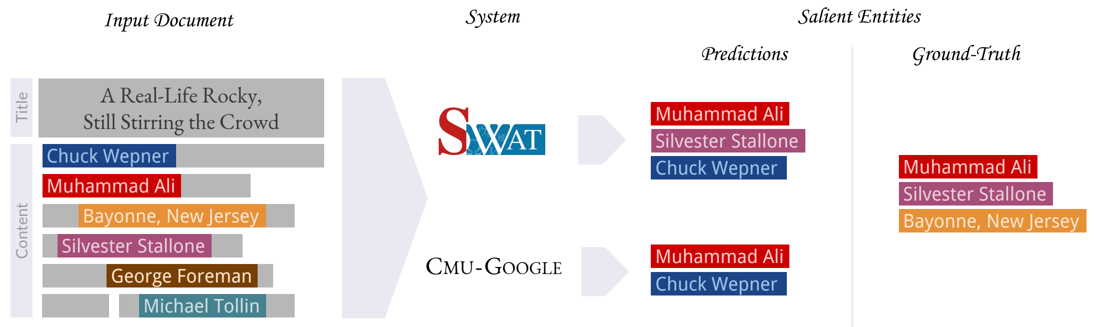


Figure 4.10: Example that shows where information present in the title help SWAT into a proper detection of salient entities.

We notice that both systems predict as salient the entities `Chuck_Wepner` and `Muhammed_Ali` that are mentioned at the beginning of the document; but, in addition, SWAT is able to correctly detect `Silvester_Stallone` as salient because it is highly related to the title which mentions `Rocky`, the movie where the actor has played as the main character.

Annotation-based Features. Features based on the scores associated with the annotations (i.e., *comm-max* and *ρ-max*) make SWAT even more robust with respect to non-coherent entities. The most interesting case for the proper understanding of the effectiveness of these features is showed in Figure 4.11. For each entity the *ρ-max* feature score is reported between parentheses.

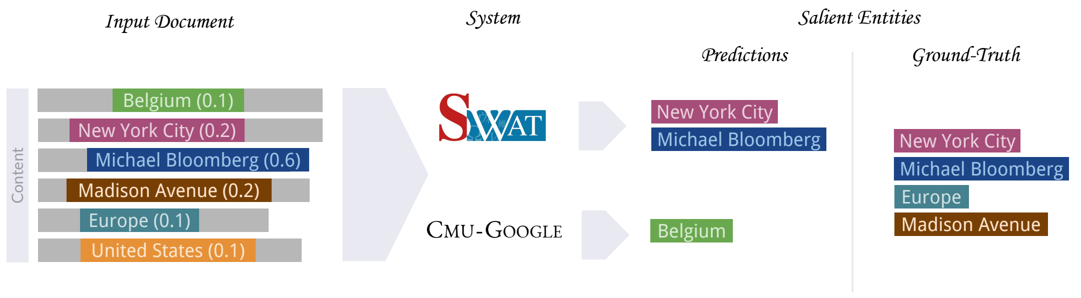


Figure 4.11: Example that shows the robustness of the SWAT's features based on the coherence of the annotations. We show between parenthesis the feature *ρ-max*.

This example shows a case where an entity can be mentioned at the beginning of the document but without being salient. Unlike CMU-GOOGLE, SWAT is robust in detecting such a kind of entities because the feature *ρ-max* gets a low score of coherence (in the text, `Belgium` is actually annotated from the adjective *"belgian"* but it is wrongly annotated as the country), which therefore allows to correctly classify it.

Relatedness-based Features. The final set of relevant features that help SWAT in performing more accurate predictions is represented by the features developed on the top of relatedness signals (i.e., *dw-cbow-pagerank/hub* features). More precisely, these features contribute to build a complete graph where nodes are the entities annotated in the input document and edges are weighted with the cosine similarity between their DeepWalk embeddings. The relatedness-based features for each entity is eventually computed by running a centrality algorithm (e.g., PageRank) over this graph. These features help SWAT in predicting as salient those entities which are central with respect to the other entities annotated in the input text, by exploiting a more sophisticated relatedness function that exploits the semantic relatedness encoded by the Wikipedia hyperlinks. This is especially useful when a salient entity is not mentioned at the beginning of the input text but it is highly related to the rest of the entities present in the input document. Figure 4.12 reports a practical example where we show the usefulness of these features, in particular of *dw-cbow-pagerank*.

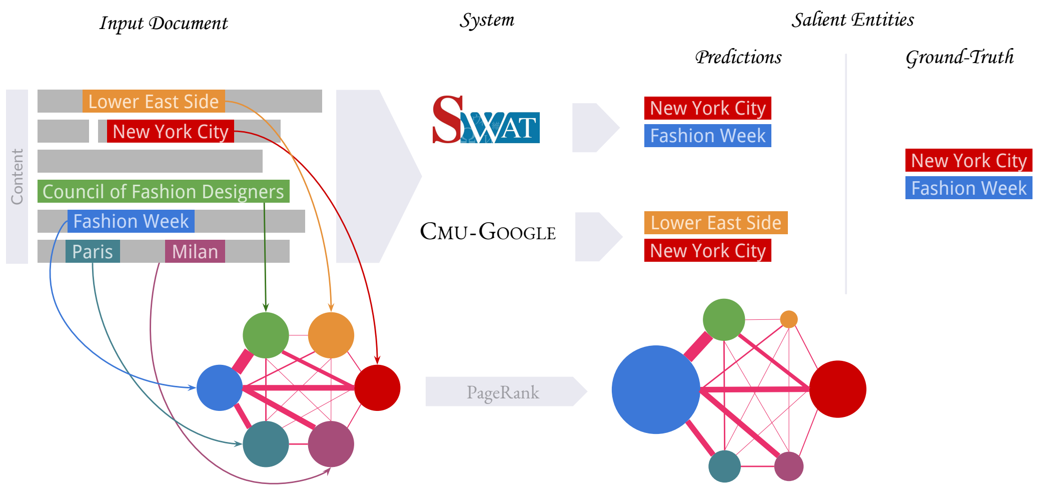


Figure 4.12: Example that shows where *dw-cbow-pagerank* feature helps SWAT in predicting the salient entities. Although several entities are not mentioned at the very beginning of the input text, they are classified as salient (resp. non-salient) because they achieve high (resp. very low) scores for *dw-cbow-pagerank*. Ticker edges mean higher DeepWalk cosine similarities between two nodes of the graph.

As we can see, both systems predict a correct salient entity that is mentioned at the beginning, namely `New_York_City`. On the other hand, CMU-GOOGLE classifies as salient also `Lower_East_Side` because it appears at the beginning, even if it is not. The reason why our SWAT does not make this error is because it takes into account how poorly related this entity is to the others.

On the other hand, SWAT correctly predicts `Fashion_Week` as salient instead of `Lower_East_Side`. By carefully looking at the computation of *dw-cbow-pagerank* feature (bottom of Figure 4.12), the node of `Fashion_Week` is linked to the others through heavy weights (ticker edges) than the ones drawn by `Lower_East_Side`. More precisely, `Fashion_Week` has a strong relatedness with `New_York_City`, `Paris` and `Milan` because they are popular fashion capitals. After the PageRank computation upon this graph, `Fashion_Week` is scored with the highest *dw-cbow-pagerank* value, whereas `Lower_East_Side` is scored much lower. Overall, SWAT is able to detect as salient entities both `New_York_City` (which appears at the beginning of the text) and `Fashion_Week` (scored with a high *dw-cbow-pagerank* value).

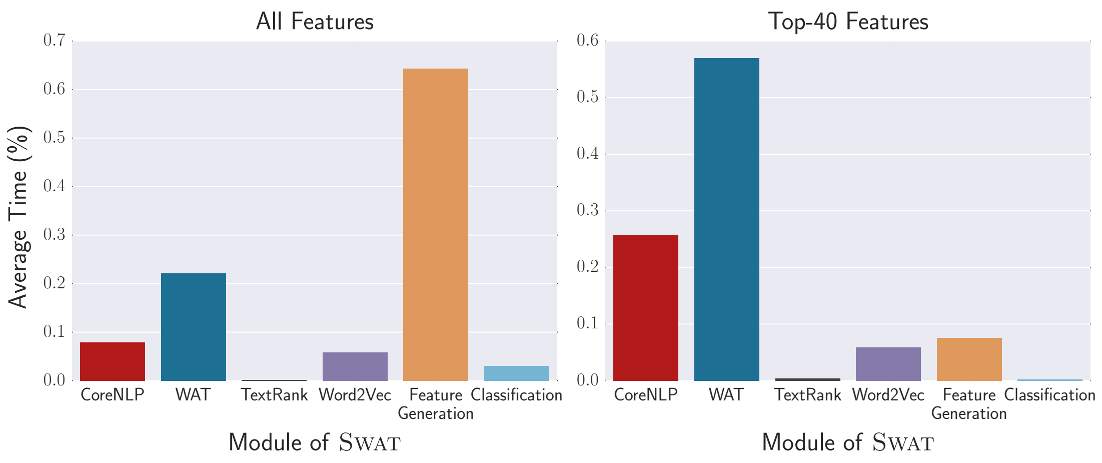


Figure 4.13: Average computation time (percentage) of each SWAT module for the whole salience-annotation pipeline by deploying all (left) and the top-40 (right) features. Performance are averaged over a sample of 400 documents of the NYT dataset.

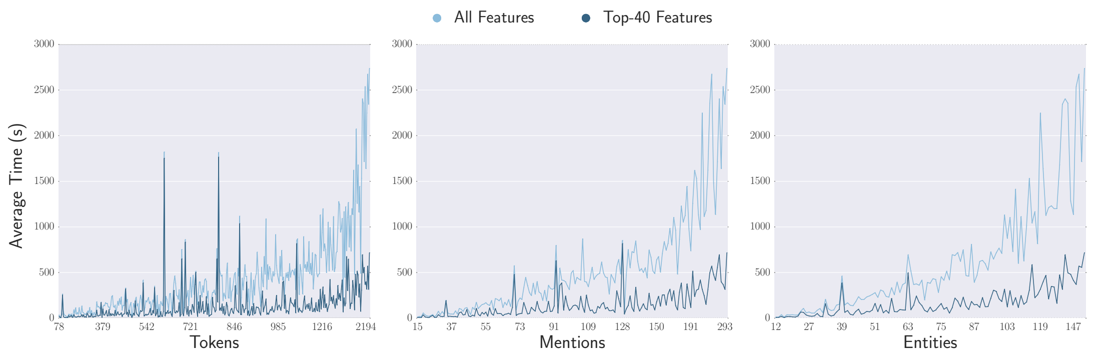


Figure 4.14: Average computation time of SWAT by distinguishing between the number of tokens, mentions and entities over a sample of 400 documents of the NYT dataset.

Time Efficiency. The average computation time of each module constituting SWAT is reported in Figure 4.13. When all features are used, the most expensive component is clearly the Feature Generation module, which takes about the 64% of the whole computation time of SWAT; whereas CoreNLP, WAT, TextRank, Word2Vec and Classification take respectively the 7%, 22%, 0.1%, 5.7% and 2% of the computation time of the whole pipeline. Conversely, when only the top-40 features learned over NYT are used, SWAT becomes much faster (up to 5×, see Figure 4.14) without any significant degradation on its accuracy (see Figure 4.8). The choice of training SWAT over NYT data is motivated by the fact that: (1) the most important

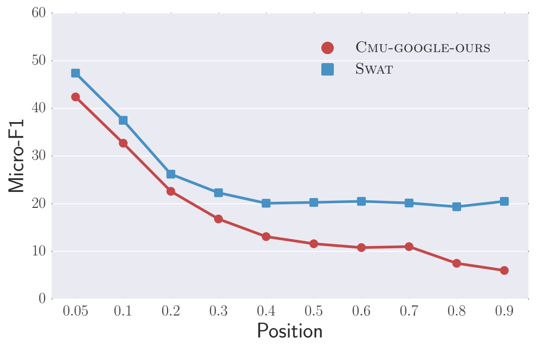


Figure 4.15: Micro-F1 performance as a function of the first token positions on the NYT dataset. Each point (x, y) indicates that the micro-F1 is y for all entities whose position is larger than x.

features are very similar to the ones derived when the system is trained on Wikinews, and (2) the system trained on NYT and then tested on Wikinews still obtains higher performance than current state-of-the-art systems.

Flexibility over Entities' Position. In this section we address a question posed by Dunietz & Gillick (2014) and concerning the evaluation of how the performance of top-systems depends on the distribution of the salient entities in the input documents. Figures 4.3–4.4 further motivate this question by showing the distribution of the salient and non-salient entities within the NYT and Wikinews datasets. As expected, most of the salient entities are concentrated on the beginning (i.e., position in the first 20%) of the news over both datasets. Moreover, the whole NYT corpus contains a significant number of them which are mentioned for the first time after the beginning of the document, with 44 192 salient entities whose first position is after the first 20% of the news for a total of 31 128 such news (out of the total 108 925 news in NYT). On the other hand, the salient entities present in Wikinews are mainly confined at the beginning of documents, with only 28 salient entities whose first position is after the first 20% of the news. For this reason we only consider NYT as the main testbed for estimating the flexibility of the systems over entities' position, both for its large size and for the wider distribution that salient entities have inside this corpus.

Figure 4.15 shows the comparison among the available systems. Performance are computed only over the test set of the NYT, which contains 3911 salient entities whose first position is after the first 20% of the news, with a total of 2821 such news (which are 9577 in total in the test set). All systems are highly effective on the classification of salient entities mentioned at the beginning of the document, but their behavior differs significantly when salient entities are mentioned at the documents' end. In this latter case, SWAT does not overfit upon the positional feature and, indeed, obtain a high improvement with respect to CMU-GOOGLE-OURS which is respectively up to 14% in micro-F1. As a consequence we can state that SWAT is more flexible with respect to salient-entities' position than CMU-GOOGLE, so that it could be used consistently over other kinds of documents where salient information is not necessarily confined to their beginning.

Error Analysis. In order to gain some insights on SWAT performance and possible improvements, we analyzed its erroneous predictions by drawing a subset of 80 $(= 40+40)$ documents from the NYT and Wikinews datasets. The most significant result we gain is what argued by Hasan & Ng (2014): namely, the deployment of semantic knowledge (i.e., Wikipedia entities) eliminates some errors that originally afflicted keyphrase extraction algorithms. However, our error analysis of 80 documents also showed that false-negative errors (i.e., entities classified as non-salient, despite being salient) are mainly due to the position-based features which frequently induce to miss a salient entity because it is not at the beginning of the news. Furthermore, we noticed that a large percentage of the analyzed news of NYT ($\sim$35%) and Wikinews ($\sim$40%) contain false-positive errors which are ground-truth errors: in these cases SWAT correctly identifies the salience of an entity, but the ground-truth does not label it as salient and so it is unfortunately counted as an error in our tables.

This analysis suggests that SWAT performance could be actually higher than what we showed before and a better ground-truth dataset should be devised, as we foresee in the concluding chapter.

4.2 FACT SALIENCE

Automatic knowledge acquisition at large scale requires the transformation of human-readable knowledge into a machine-understandable format. Machine-readable information is usually structured in the form of facts, in which a given relation links a set of arguments — e.g., *("US", "withdraws from", "Iran nuclear deal").* Facts are at the core of several natural language understanding applications such as knowledge graph construction (Nguyen et al., 2017a), question answering (Abujabal et al., 2018), structured search (Bast et al., 2014), or entity linking (Cheng & Roth, 2013). Different approaches aim to discover facts from natural language text. In the extremes of the spectrum, relation extraction (Mintz et al., 2009) looks for all facts linkable to a KG, whereas open information extraction (Banko et al., 2007) extracts facts over an unconstrained set of arguments from an input text, without link them to any entity or relation present in a KG. In this thesis, we aim to additionally score facts (extracted via open information extraction) according to their prominence.

We define *fact salience* as the task of discovering the most prominent open facts in a text document. A fact is salient if it carries the essential information that the text conveys. A higher salient score denotes higher prominence, determining a ranking across all facts in the document. This ranking must reflect relevance and diversity: we want the top-*k* facts to compress the most relevant information in the smallest number of facts.

Here we present SalIE (**Sal**ient **I**nformation **E**xtraction), the first fact salience system able to output a ranking of *salient open facts* from a text document. SalIE is unsupervised and knowledge agnostic. It uses facts as atomic units and PageRank to detect their relevance. It also exploits the fact structure to promote diversity via clustering.

We evaluated SalIE on a real-world dataset and compared it with the strong positional baseline (facts appearing first are more relevant) and with two top text summarizers (one reimplemented to work at fact level). SalIE outperforms baselines and text summarization competitors particularly when the size of the output is restricted, suggesting that facts, as atomic units expressing a single proposition (Del Corro & Gemulla, 2013), are an effective way to compress information.

The source code and the processed datasets are publicly available[*] to encourage further developments of the fact salience task.

4.2.1 More on Fact Salience

Fact salience is the task of extracting *salient facts* from a text document. Salient facts must fulfil two requirements: (1) *relevance* and (2) *diversity*. A fact is relevant if it carries the essential information that the text conveys. A fact is not relevant per se but in a specific context.

[*]github.com/mponza/SalIE

In an article about the US-Iran nuclear deal the fact *("US", "withdraws from", "Iran Nuclear Deal")* is more relevant that *("Washington", "is", "US capital")*. The output of a fact salience system must ensure that the top-k facts contain the maximum information in the smallest number of facts. This implies a dependency between facts, as less relevant facts should be penalized when they carry information already contained in more relevant ones.

4.2.2 Our Proposal: SalIE

SalIE is a graph-based method for the extraction of *salient* open facts in text documents. Open facts are a structured machine-readable representation of the information in text. Its arguments are not linked to an existing KG. SalIE takes as input all open facts detected by an open information extraction system — in our implementation we use MinIE (Gashteovski et al., 2017).

SalIE works in two stages: (1) relevance and (2) diversification. First, a graph with open facts as nodes is instantiated so that PageRank assesses their relative relevance. Later, a clustering algorithm selects a diversified set of facts.

Fact Relevance

SalIE computes fact *relevance* by growing a complete graph of open facts $G_{OF} = (V, E)$ extracted from the input text. *Coherence* is induced by weighting the edges E between nodes V, whereas a *relevance* prior is induced via the instantiation of the PageRank's teleport vector.

Step 1 – Facts as Nodes. Each node is a fact extracted by MinIE. Undefined facts with a no clear co-reference — e.g. *("He", "plays", "softball")* — or facts with constituents composed by single words that are generally uninformative or noisy — e.g. *("doorman", "has", "age")* — are removed.

Step 2 – Coherence: Edge Weighting. We want related facts to get a higher weight assuming that the most relevant facts will be those more central. We weight each edge (u, v) with the *semantic similarity* between u and v as the cosine between the centroid of the word embeddings in the facts. Stanovsky et al. (2015) have shown that learning word embeddings with open facts allows the generation of higher quality vectors . The assumption is that the relatedness of words within a fact is stronger than with words outside. This provides the basis for more accurate contextualization. Accordingly, in our implementation we use GloVe (Pennington et al., 2014) trained on the Wikipedia corpus using open facts extracted by MinIE for co-occurence context.

Step 3 – Relevance Prior. We introduce a prior for each fact by computing a score used to instantiate the PageRank's teleport vector. The assumption is that authors tend to express the most relevant facts at the beginning. We instantiated each fact teleport as $factPrior(i) = x_i / ||X||$, where $x_i = |V| - i$ and i is the fact index. This is important especially for news where the lead paragraph is the most important part of the article. That's why the positional baseline is so strong in tasks as text summarization or entity salience (Ponza et al., 2017b, 2018b).

Step 4 – Relevance Computation. This stage runs PageRank on the graph. The stationary distribution will capture the *relevance* of each open fact. This distribution reflects the semantic centrality of each fact weighted by its relevance prior.

Fact Diversification

In this stage SalIE diversifies the set of *relevant* facts computed in the previous stage. Facts are clustered exploiting the fact structure, and the most relevant facts in each cluster are selected according to the relevance scores.

Facts have clear semantics regarding the role of each of its constituents (i.e., subject, relation, and object) in the proposition. SalIE exploits this by clustering together those facts that have the same head in the subject's constituent. As the subject is typically the theme (or topic) of the clause (Quirk et al., 1985), the intuition here is that facts with the same subject express information about the same entity. Therefore, each cluster will contain a ranked set of facts about each entity in the document.

After the facts have been clustered, we iteratively select facts from each cluster according to its relevance until we reach the desired number of facts as output. The number of facts in the output is a parameter of the system.

4.2.3 Experiments

In this section we evaluate our system SalIE on an experimental assessment performed on a large and real-world dataset, where we compare our system with three different baselines and with two top text summarizers.

Given a document we want to evaluate how salient the top-k facts are. As the number of facts in the ranking is a parameter of the model, we evaluate 5 configurations: from top-1 to top-5 facts.

Datasets

As there is no dataset to directly asses the saliency of facts, we compare the extracted facts in each ranking with a manually generated summary. We use the New York Times (Sandhaus, 2008) corpus, consisting of 3956 news articles and summaries from 2007 (with summaries larger than 50 tokens) as described by Durrett et al. (2016).

Evaluation Metrics

To measure how close is the ranking to the summary, we use the ROUGE package[*], standard for document summarization (Lin, 2004). Rouge-1 measures the presence of single words between the salient facts and the summary; ROUGE-L identifies the longest common subsequence (LCS) with maximum length between facts and summary; ROUGE-1.2W measures the weighted LCS by taking into account spatial relations and giving higher values to consecutive matches; ROUGE-SU is the number of occurring bigrams between the facts and summary with arbitrary gaps. For each metric we report the F1 performance, all computed with a 95% confidence interval, run with stemming and stopword removal[†].

Note that we do not take into account the correctness of the facts (i.e., if they are well-structured). All systems implemented, except the Berkeley summarizer (Durrett et al., 2016), use the same open facts extracted by MinIE. Also for the Berkeley summarizer, we do not evaluate the structure or fluency of the summary.

[*] pypi.org/project/pyrouge/0.1.3
[†] Executing package arguments: `-c 95 -m -s -U -w 1.2`.

Systems and Baselines

SalIE. It outputs top-k salient facts per article. We show results for two MinIE configurations: *safe* and *aggressive*, which differ in the fact average size.

Baselines. As there is no direct fact salience competitor, we designed three baselines: The standard Position baseline which ranks facts with respect to their order of appearance, TF-IDF which ranks them with respect to the subject's head TF-IDF and the Context baseline which ranks facts with respect to the cosine similarity between the document and the fact embedding's centroid.

Document Summarizers. We used two state-of-the-art document summarizers, i.e. the unsupervised graph-based TextRank (Mihalcea & Tarau, 2004) and the supervised Berkeley summarizer (Durrett et al., 2016). We adapted TextRank to work with facts instead of sentences. For the Berkeley summarizer, we used the model online[*]. As the size of the summaries is a parameter of the summarizer, we set it to match the average size of MinIE facts (*safe* is 10 and *aggressive* is 6), For example, for the top-5 configuration in the aggressive mode, the summary length is set to 30.

Tables 4.9 and 4.10 shows examples for the position baseline, the text summarizers and SalIE.

Results and Analysis

Tables 4.11 and 4.12 show the results for all the systems and baselines. We use colors **black**, **gray** and **light gray** for the first, second and third best performing methods. In each ROUGE configuration, we show results for five rankings: top-1 to top-5. Differences between SalIE and the best competitor is reported in the last line of the tables.

Table 4.11 shows the results where facts have been extracted with MinIE's *safe* mode. SalIE outperforms all other methods and baselines for the first three rankings (top-1 to top-3), although Berkeley summarizer comes first in top-4 and 5 facts as a higher budget takes the system closer to the gold standard human-readable summaries. TextRank has an opposite behaviour compared to Berkeley, performing well in top-1 and 2 but lagging behind as more

[*]nlp.cs.berkeley.edu/projects/summarizer.shtml

Table 4.9: MinIE *safe* mode.

Human Summary		
Body of Toni Grossi Abrams, widow and Staten Island socialite, is found in warehouse on outskirts of Panama City, Panama, where she had moved to begin career in real estate; Debra Ann Ridgley, one of her tenants, is charged with stabbing Abrams to death in her apartment on April 9.		
Method		**Salient Facts / Summary**
Position	1	*("Surgery patients", "lie low in", "style retreat")*
	2	*("Remains", "were discovered beside warehouse at", "edge of cinder-topped soccer field on outskirts of Panama City")*
	3	*("Abrams", "had been stabbed to death in", "apartment")*
TextRank	1	*("Ridgley", "was in Abrams's apartment", "Garcia and friend")*
	2	*("Ridgley", "was in Abrams's apartment that", "night")*
	3	*("Abrams's body", "remains in", "Panama City morgue")*
Berkeley		*"The widow of a mortgage executive, Ms. Abrams was something of a force of nature in Staten Island society. The suspect, Debra Ann Ridgley, is."*
SalIE	1	*("Abrams", "had been stabbed to death in", "apartment")*
	2	*("Remains", "were discovered beside warehouse at", "edge of cinder-topped soccer field on outskirts of Panama City")*
	3	*("Apartment", "tending wounds at time of", "murder")*

Table 4.10: MinIE *aggressive* mode.

Human Summary		
Russian state oil company Rosneft has lined up $22 billion in financing from consortium of Western banks to buy assets from bankrupt rival Yukos; Rosneft says it will bid for refineries owned by Yukos as outlet for production from its Yugansk subsidiary in western Siberia; some of banks listed.		
Method		**Salient Facts / Summary**
Position	1	*("State oil company", "lined up $ from consortium of banks buy assets from", "rival")*
	2	*("Rosneft", "increase footprint in", "oil and gas business")*
	3	*("Bids", "are successful as", "expected")*
TextRank	1	*("Banks", "made loans to", "Rosneft and state company")*
	2	*("Banks", "lent company related to Rosneft", "$ increase share")*
	3	*("State oil company", "lined up $ from consortium of banks buy assets from", "rival")*
Berkeley		*"The Russian state oil company Rosneft has lined up $22 billion from a consortium of Western banks."*
SalIE	1	*("State oil company", "lined up $ from consortium of banks buy assets from", "rival")*
	2	*("Banks", "made loans to", "Rosneft and state company")*
	3	*("Rosneft", "increase footprint in", "oil and gas business")*

facts are added probably due to the lack of a diversification stage. It is interesting to note that systems working at the fact level do well in constrained settings, suggesting that facts may be an effective way to compress information.

Table 4.12 shows the results when MinIE is used in *aggressive* mode. In this experiment, we aim at analyzing the flexibility of the systems when applied in a scenario when they need to rank very short facts or span of texts. As we can see, SalIE achieves the highest performance overall metrics independently the number of facts used, with the only exception of the ROUGE-1.2W and ROUGE-SU score when 4 or 5 facts are used. The second and third best performing methods are Position and TextRank. Again, in this case, it is suggested that facts are an appropriate mechanism to compress information.

Overall SalIE shows a more stable balance across all rankings in both settings. It always ranks first or second (except in ROUGE-SU top-5 where it comes third). Compared to TextRank it seems to significantly better manage redundancy, while compared to the Berkeley it does better at detecting relevant information in constrained settings. This is due to the use of facts for compressing.

Table 4.11: MinIE *safe* mode.

Method	ROUGE-1					ROUGE-L					ROUGE-1.2W					ROUGE-SU				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
Position	13.9	20.4	24.8	27.8	29.7	12.8	18.1	21.8	24.4	26.0	6.20	8.80	10.8	12.2	13.2	2.70	5.30	7.50	9.00	10.0
TF-IDF	10.5	15.8	19.0	21.0	22.3	9.60	13.2	15.7	17.5	18.5	4.60	6.30	7.50	8.50	9.20	1.40	2.80	4.00	4.80	5.30
Context	13.6	19.6	22.8	24.6	25.7	11.5	16.3	18.9	20.4	21.3	5.60	8.00	9.40	10.3	11.0	2.50	4.40	5.60	6.30	6.70
TextRank	15.2	21.5	24.5	26.1	26.8	13.0	17.5	19.8	21.3	22.0	6.20	8.40	9.70	10.6	11.2	2.60	4.90	6.40	7.20	7.50
Berkeley	8.50	18.0	25.4	30.4	34.1	8.00	16.3	22.5	26.7	29.7	3.80	7.70	11.0	13.2	14.9	0.80	3.40	6.90	10.1	12.7
SalIE	17.1	24.2	28.0	30.0	30.9	15.3	21.2	24.3	26.0	26.8	7.40	10.3	12.0	13.1	13.6	3.60	6.50	8.30	9.20	9.50
Diff.	**+1.9**	**+2.7**	**+2.6**	−0.4	−3.2	**+2.3**	**+3.1**	**+1.8**	−0.7	−2.9	**+1.2**	**+1.5**	**+1.0**	−0.1	−1.3	**+0.9**	**+1.2**	**+0.8**	−0.9	−3.2

Table 4.12: MinIE *aggressive* mode.

Method	ROUGE-1					ROUGE-L					ROUGE-1.2W					ROUGE-SU				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
Position	10.1	15.3	19.3	22.3	24.5	9.60	13.8	17.2	19.8	21.7	4.40	6.30	7.80	9.10	10.1	1.40	2.90	4.50	5.80	6.90
TF-DF	8.90	13.6	16.4	18.3	19.6	8.30	11.3	13.4	15.0	16.1	3.80	5.20	6.20	6.90	7.60	0.90	2.00	2.90	3.50	4.10
Context	9.50	14.5	17.9	20.0	21.4	8.40	12.4	15.1	16.8	18.0	3.90	5.70	6.90	7.80	8.50	1.20	2.40	3.40	4.20	4.80
TextRank	11.3	17.2	20.3	22.2	23.3	10.1	14.3	16.7	18.2	19.2	4.60	6.40	7.60	8.40	9.00	1.30	2.90	4.20	5.00	5.60
Berkeley	3.60	10.6	16.2	21.2	25.4	3.50	9.90	14.8	19.0	22.5	1.60	4.60	7.00	9.10	11.0	0.20	1.20	2.80	4.80	6.90
SalIE	11.6	17.9	21.6	24.2	25.9	10.5	15.9	19.1	21.3	22.8	4.80	7.20	8.60	9.70	10.5	1.60	3.30	4.60	5.70	6.50
Diff.	**+0.3**	**+0.7**	**+1.3**	**+1.9**	**+0.5**	**+0.4**	**+1.6**	**+1.9**	**+1.5**	**+0.3**	**+0.2**	**+0.8**	**+0.8**	**+0.6**	−0.5	**+0.2**	**+0.4**	**+0.1**	−0.1	−0.4

Part III
Applications

5
Algorithms for Expert Finding

Searching the human expertise has recently attracted considerable attention in the Information Retrieval community. This is a computationally challenging task because *human expertise* is hard to formalize. Literature often refers to human expertise as a sort of "tacit knowledge": the expertise carried by people belongs to their minds and thus difficult to be extracted as well as to be modeled. As a consequence, a system has one way to assess and access the expertise of a person: through artifacts of the so-called "explicit knowledge", i.e., something that is already captured, documented, stored via, for example, documents.

In this chapter, we present Wiser, a new solution for solving the problem of *Expert Finding*, namely the retrieval of pertinent people (i.e., experts) with respect to a given input query. Our approach is unsupervised and it jointly combines classical language modeling techniques, based on text evidences, with the Wikipedia knowledge graph, through the extraction of textual knowledge via entity linking.

Wiser indexes each academic author through a novel profiling technique which models her expertise with a small, labeled and weighted graph drawn from Wikipedia. At query time, experts are retrieved by combining classic document-centric approaches, which exploit the occurrences of query terms in the author's documents, with a novel set of profile-centric scoring strategies, which compute the relatedness between the author's expertise and the query topic via the above graph-based profiles. The effectiveness of our system is established over a large scale experimental test on a standard dataset for this task. We show that WISER achieves better performance than all the other competitors, thus proving the effectiveness of modelling author's profile via our graph of entities.

5.1 Introduction

Research on how to provide a way to share expertise can be traced back to at least the 1960s (Brittain, 1975). In more recent years, the explosion of digital information has revamped the scientific interest on this problem and led researchers to study and design software systems that, given a topic X, could support the *automatic search* for candidates with the expertise X. Initial approaches were mainly technical and focused on how to unify disparate and dissimilar document collections and databases into a single data warehouse that could easily be mined. They employed some heuristics or even required people to self-judge their skills against a predefined set of keywords (Ackerman et al., 2002; Yimam & Kobsa, 2000). Subsequent approaches have been proposed to exploit techniques proper of document retrieval, and they have been applied to the documents written by or associated to each expert candidate as the main evidence of her expertise (Balog et al., 2012). However, classical search engine return documents not people or topics (Heath et al., 2006).

Today they do exist advanced systems which may be classified into two main categories: *expert finding systems*, which help to find who is an expert on some topic, and *expert profiling systems*, which help to find of which topic a person is an expert. Balog et al. (2012) summarize the general frameworks that have been used to solve these two tasks (see also Section 5.2), and look at them as two sides of the same coin: an author retrieved as expert of a topic should contain that topic in her profile. However, as pointed out by Van Gysel et al. (2016a,b), known systems are yet poor in addressing three key challenges which, in turn, limit their efficiency and applicability (Balog et al., 2012; Li et al., 2014): (1) queries and documents use different representations so that maximum-likelihood language models are often inappropriate, and thus there is the need to make use of semantic similarities between words; (2) the acceleration of data availability calls for the further development of unsupervised methods; (3) in some approaches, a language model is constructed for every document in the collection thus requiring to match each query term against every document.

In this chapter we focus on the task of *experts finding in the academia domain*, namely, we wish to retrieve academic authors whose expertise is defined through the publications they wrote and it is relevant for a user query.

In this context, the best system to date is the one recently proposed by Van Gysel et al. (2016b). It has a strong emphasis on unsupervised profile construction, efficient query capabilities and *semantic matching* between query terms and candidate profiles. Van Gysel et al. have shown that their unsupervised approach improves retrieval performance of vector space-based and generative-language models, mainly due to its ability to learn a *profile-centric* latent representation of academic experts from their publications. Their key idea is to deploy an embedding representation of words — such as the one proposed in (Mikolov et al., 2013b) — to map conceptually similar phrases into geometrically *close* vectors (i.e., *"nyt"* is mapped into a vector close to the one of *"New York Times"*). At query time, their system first maps the user query into the same latent space of experts' profiles and, then, retrieves the experts showing the highest dot-product between the embeddings of their profile and the one of the query. This way the system can efficiently address the *mismatch problem* between the "language" of the user query and the "language" of authors' documents: i.e., an expert can be identified even if her documents do not contain any terms of the input query (Li et al., 2014).

But, despite these recent improvements, the *semantic matching* implemented by Van Gysel et al. (2016b) is yet limited to the use of *latent* concepts, namely ones that cannot be explicitly defined and thus cannot *explain* the why an expert profile matches a user query. In this work we propose a novel approach for expert finding which is still unsupervised but, unlike Van Gysel et al. (2016b), takes advantage of the recent IR trends in the deployment of knowledge graphs (Dietz et al., 2017; Weikum et al., 2016) which allow modern search engines and NLP/IR tools to be more powerful in *semantically matching* queries to documents and allow to *explicitly represent concepts* occurring in those documents, as well-defined nodes in these graphs. More specifically, our approach models the academic expertise of a researcher both syntactically and semantically by orchestrating a document-centric approach, that deploys an open-source search engine (namely Elasticsearch), and a profile-centric approach, that models in an innovative way the individual expert's knowledge not just as a list of words or a vector of latent concepts — as in (Van Gysel et al., 2016b) — but as a small *labeled and weighted graph* derived from Wikipedia, which is the best known and open knowledge graph to date. That graph will consist of labeled nodes, which are the entities mentioned in author's publications — detected via TagMe (Ferragina & Scaiella, 2012), one of the most effective entity linking system to date — and edges weighted by means of proper entity relatedness scores — computed via an advanced framework (Ponza et al., 2017a). Moreover, every node is labeled with a *relevance score* which models the pertinence of the corresponding entity to author's expertise, and is computed by means of proper random walk calculation over the author's graph; and with a *latent vector representation* which is learned via entity and other kinds of structural embeddings, that are derived from Wikipedia and result different from the ones proposed in (Van Gysel et al., 2016b). The use of this enriched graph allows to obtain a finer, explicit and more sophisticate modeling of author's expertise that is then used at query time to search and rank experts based on the *semantic relation* that exist between the words/entities occurring in the user query and the ones occurring in the author's graph.

This novel modelling and querying approach has been implemented in a system called Wiser, which has been experimented on the largest available dataset for benchmarking academia expert finding systems, namely TU dataset (Berendsen et al., 2013). This dataset consists of a total of 31 209 documents, authored by 977 researchers, and 1266 test queries with a human-assessed ground-truth that assigns to each query a ranking of its best academic experts. Wiser shows statistically significant improvements over different ranking metrics and configurations. More precisely, our document-centric approach improves the profile-centric Log-linear model proposed by Van Gysel et al. (2016b) of +7.6%, +7.4% and +7% over MAP, MRR and NDCG scores. Whereas our profile-centric approach based on entity linking improves that Log-linear model of +2.4% in MAP, and achieves comparable results for the other metrics. Then, we show that a proper combination of our document- and profile-centric approaches achieves a further improvement over the Log-linear model of +9.7%, +12.6% and +9.1% in MAP, in MRR and in NDCG; and, furthermore, it improves the sophisticated Ensemble method of Van Gysel et al. (2016b), which is currently the state-of-the-art, of +5.4%, +5.7% and +3.9% on MAP, MRR and NDCG@100 metrics, respectively. This means that Wiser is designed upon the best *single model* and the best *combined models* today, thus resulting the state-of-the-art for the expert finding problem in academia.

A publicly available version of Wiser is available [*] for testing its functionalities about expert finding and expert profiling over the researchers of the University of Pisa.

The next sections will review the main literature about expert finding solutions (Section 5.2), in order to contextualize our problem and contributions; describe the design of Wiser, by detailing its constituting modules and their underlying algorithmic motivations (Section 5.4) and finally presenting our achievements through a large set of experiments (Section 5.5).

5.2 Related Work

We first discuss prior work on experts finding by describing the main challenges of this task and its differences with classic document retrieval. Then we move on to describe how our work differs from known experts finding (and profiling) approaches by commenting about its novel use of entity linking, relatedness measures and word/entity embeddings. Finally, in the last part of this section, we will concentrate on detailing the main differences between Wiser and the state-of-the-art system proposed by Van Gysel et al. (2016b), because it is also the most similar to ours.

Experts finding systems differ from classic search engines (Chakrabarti, 2002; Manning et al., 2008) in that they address the problem of finding the right *person* (in contrast with the right document) with appropriate skills and knowledge specified via a user query. Preliminary attempts were made in adapting classic search engines to this task with poor results (Balog et al., 2012). The key issue to solve is how to represent the individual expert's knowledge (Balog et al., 2006, 2009; Macdonald & Ounis, 2006; Van Gysel et al., 2016b). Among the several attempts, the ones that got most attention and success were the *profile-centric* models (Balog et al., 2006; Van Gysel et al., 2016b) and the *document-centric* models (Balog et al., 2012; Cao et al., 2005; Macdonald & Ounis, 2006). The first ones work by creating a profile for each candidate according to the documents they are associated with, and then by ranking experts through a matching between the input query and their profiles. The second ones work by first retrieving documents which are relevant to the input query and then by ranking experts according to the relevance scores of their matching documents. The joint combina-

[*] wiser.d4science.org

tion of these two approaches has shown recently to further improve the achievable performance (Balog & De Rijke, 2008; Van Gysel et al., 2016b), as we will discuss further below.

Most of the solutions present in the literature are *unsupervised* (Balog et al., 2006, 2009; Cao et al., 2005; Macdonald & Ounis, 2006; Van Gysel et al., 2016b) since they do not need any training data for the deployment of their models. *Supervised* approaches (Macdonald & Ounis, 2011; Moreira et al., 2015) have been also proposed, but their application has usually been confined to data collections in which query-expert pairs are available for training (Fang et al., 2010; Sorg & Cimiano, 2011). This is clearly a limitation that has indeed led researchers to concentrate mainly onto unsupervised approaches.

The focus of our work is onto the design of *unsupervised academia experts finding* solutions which aim at retrieving experts (i.e., academic authors) whose expertise is properly defined through the publications they wrote. Among the most popular *academic* expert finding solutions we have ArnetMiner (Tang et al., 2008), a system for mining academic social networks which automatically crawls and indexes research papers from the Web. Its technology relies on a probabilistic framework based on topic modeling for addressing both author ambiguity and expert ranking. Unfortunately, the implementation of the system is not publicly available and it has not been experimented on publicly available datasets. Similar comments hold true for the Scival system by Elsevier.[*]

Among the publicly available systems for academia expert finding, the state-of-the-art is the one recently proposed by Van Gysel et al. (2016b). It adapts a collection of unsupervised *neural-based* retrieval algorithms (Van Gysel et al., 2017), originally deployed on product search (Van Gysel et al., 2016a), to the experts finding context via a log-linear model which learns a *profile-centric* latent representation of academic experts from the dataset at hand. At query time, the retrieval of experts is computed by first mapping the user query into the same latent space of experts profiles and, then, by retrieving the experts with the highest dot-product between their profile and the query.

To the best of our knowledge we are the first to design an experts finding system for the academia domain which is based on entity linking and embeddings techniques built upon the Wikipedia KG (Balog et al., 2012; Van Gysel et al., 2016b). The key feature of our sys-

[*]See scival.com.

tem WISER is a *novel profile model for academic experts*, called Wikipedia Expertise Model, that deploys those advanced techniques to build a small *labeled and weighted graph* for each academic author. This graph will describe her individual "explicit" knowledge in terms of Wikipedia entities occurring in her publications and of their relatedness scores computed by means of Wikipedia-based interconnections and embeddings. This graph representation is then used at query time to efficiently search and rank academic experts based on the "semantic" relation that exists between their graph model and the words and entities occurring in the user query.

5.3 Notation and Terminology

A dataset (D, A) for the experts finding problem is a pair consisting of a set of documents $d \in D$ and a set of authors (candidate experts) $a \in A$. We indicate with D_a the set of documents written by author a.

Entities are annotated in texts (both documents and queries) through the entity linker TAGME (Ferragina & Scaiella, 2012), which also provides a confidence score ρ_e which expresses the semantic coherence between entity e and its surrounding text in the input document. Since an entity e can be mentioned many times in the documents of a, with possibly different values for ρ_e, we denote by $\rho_{e,a}$ the *maximum* confidence score among all occurrences of e in D_a's documents. We use E_a to denote the set of all entities annotated in the documents D_a of author a.

Given an entity e, we use A_e to denote the set of authors who mention e in one of their documents, D_e to denote the subset of documents that mention e, and $D_{a,e}$ to denote the subset of documents written by author a and which mention e.

A generic input query is indicated with q, E_q will be used to denote the set of entities annotated in q by TAGME and $D_{a,q}$ will be used to denote the subset of documents D_a which are (syntactically or semantically) matched by the input query q.

5.4 Our New Proposal: Wiser

In this section we describe Wiser, whose name stands for **Wi**kipedia Experti**se** **R**anking. It is a system for academia experts finding, built on top of three main tools:

- TagMe (Ferragina & Scaiella, 2012), a state-of-the-art entity linker for annotating Wikipedia pages mentioned in an input text;
- Elasticsearch[*], an open-source software library for the full-text indexing of large data collections;
- WikipediaRelatedness (Ponza et al., 2017a), a framework for the computation of several relatedness measures between Wikipedia entities and whose suite of algorithms has been introduced in Chapter 3.

By properly orchestrating and enriching the results returned by the above three tools, Wiser offers both document-centric and profile-centric strategies for solving the experts finding problem, thus taking advantage of the positive features of both approaches. More specifically, Wiser first builds a document-centric model of the explicit knowledge of academic experts via classic document indexing (by means of Elasticsearch) and entity annotation (by means of TagMe) of the authors' publications. Then, it derives a novel profile-centric model for each author that consists of a small, labeled and weighted graph drawn from Wikipedia. Nodes in this graph are the entities mentioned in the author's publications, whereas the weighted edges express the semantic relatedness among these entities, computed via WikipediaRelatedness. Every node is labeled with a *relevance score* which models the pertinence of the corresponding entity to author's expertise, and is computed by means of proper random walk calculation over that graph; and with a *latent vector representation* which is learned via entity and other kinds of structural embeddings derived from Wikipedia. This graph-based model is called **W**ikipedia **E**xpertise **M**odel (wem) of an academic author (details in Section 5.4.1).

At query time, Wiser uses proper data fusion techniques (Macdonald & Ounis, 2006) to combine several authors' ranking: the one derived from the documents' ranking provided by

[*] www.elastic.co

Elasticsearch, and others derived by means of properly defined "semantic matchings" between the query and the Wikipedia Expertise Model of each author. This way, it obtains a unique ranking of the academic experts that captures syntactically and semantically the searched expertise within the "explicit knowledge" of authors (details in Section 5.4.2).

The following sections will detail the specialties of our novel Wikipedia Expertise Model, and its construction and use in the two phases above.

5.4.1 Data Indexing and Experts Modeling

This is an off-line phase which consists of two main sub-phases whose goal is to construct the novel *Wikipedia Expertise Model* for each academic author to be indexed. A pictorial description of this phase is provided in Figure 5.1.

Data Acquisition. In this first sub-phase, WISER indexes the authors' publications by means of Elasticsearch and annotates them with Wikipedia's entities by means of TAGME. For each input document, Elasticsearch stores information about its author a and its textual content, whereas TAGME extracts the Wikipedia entities e that are mentioned in the document together with their ρ-score that, we recall, captures the coherence between the annotated entity and the surrounding textual context in which it has been mentioned. Given that the annotated documents are scientific publications, they are well written and formatted so that TAGME is very effective in its task of extracting relevant Wikipedia entities. Subsequently, WISER filters out the entities e such that $\rho_{e,a} \leq 0.2$ (as suggested by the TAGME's documentation), since those entities are usually noisy or non coherent with the topics mentioned in the annotated document. Eventually, all this information is stored in a MongoDB[*] database.

Wikipedia Expertise Model (WEM). In this second sub-phase, WISER creates an innovative profile of each academic author that consists of a *graph* whose nodes are labeled with the Wikipedia entities found in author's documents, and whose edges are weighted by deploying entity embeddings and the structure of the Wikipedia graph, by means of the WikipediaRelatedness framework. More precisely, the expertise of each author a is modeled as a labeled and weighted graph $G_a = (V, E)$ where each node $u \in V$ is a Wikipedia entity annotated in

[*] www.mongodb.com

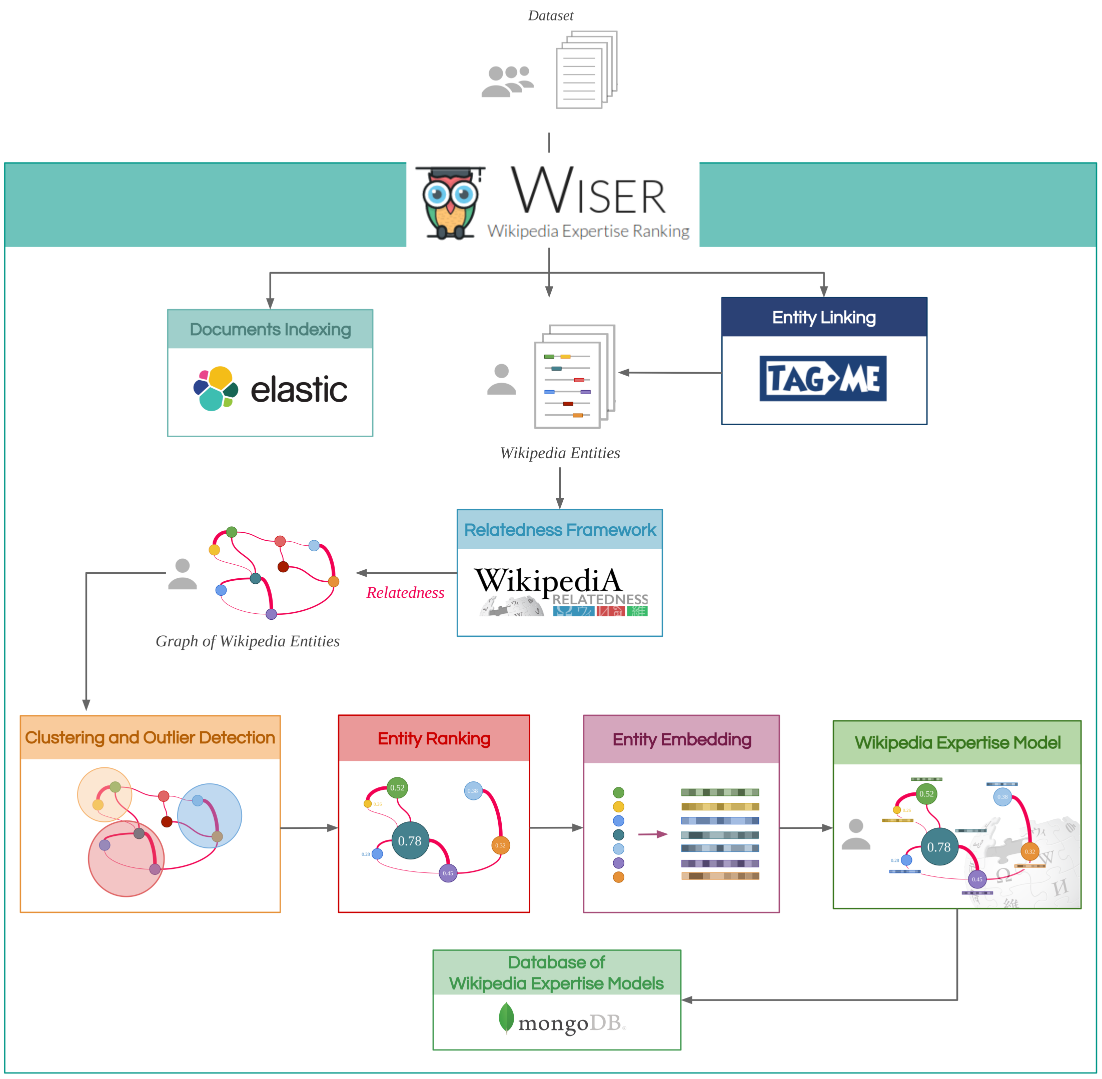


Figure 5.1: The construction for a given author of the Wikipedia Expertise Model in WISER.

at least one of the documents of D_a by TAGME, and each weighted edge $(u, v) \in E$ models the relatedness between the two entities u and v. In our context we weight (u, v) by computing the Milne&Witten relatedness measure between u's and v's entity, using the WikipediaRelatedness framework. This measure has shown its robustness and effectiveness in different domains (Ferragina & Scaiella, 2012; Ponza et al., 2017a; Scaiella et al., 2012), we leave the use of more sophisticated relatedness measures, present in WikipediaRelatedness (Ponza et al., 2017a), to a future work.

The graph G_a is further refined by executing an outlier-elimination process performed via a *graph clustering* algorithm that recognizes and removes from G_a those entities that do not belongs to any cluster and thus can be considered as off-topic for the author a. For this task WISER deploys HDBSCAN (McInnes & Healy, 2017), a density-based hierarchical clustering method based on the classic DBSCAN (Manning et al., 2008). The choice in the use HDBSCAN is motivated by its efficiency and a higher clustering quality than other popular algorithms (i.e., k-means) (McInnes & Healy, 2017). As in any clustering algorithm, input parameters of HDBSCAN strongly depend on the input graph and its expected output. In our experiments we observed that sometimes the entities labeled as outliers are not much off-topic (false positives), while in other cases no outliers are detected although they do exist at a human inspection (false negatives). WISER deals with those issues by adopting a conservative approach: if more than 20% of the nodes in G_a are marked as outliers, we consider the output provided by HDBSCAN as not valid, and thus we keep all nodes in G_a as valid topics for the examined author a.

After the application of the outlier-elimination process, WISER computes *two attributes* for each node (hence, entity) in the graph G_a. The first one is the PageRank's relevance score of an entity e mentioned by the author a. This score is computed by running the PageRank algorithm over the graph G_a with a proper setting of the damping factor to 0.85, as commonly chosen in literature (Boldi & Vigna, 2014). Moreover, the starting and teleporting distributions over G_a's nodes are defined to reflect the number of times author a mentions the entity e assigned to that node, and it is scaled by the ρ-score that evaluates how much that entity is reliable as a's research topic according to TAGME: namely, $Pr(e) = \frac{\rho_{e,a}}{C} \log(1 + |D_{a,e}|)$. Constant C is a normalization factor that makes that formula a probability distribution over the entities labeling the nodes of G_a. This definition allows the more frequent and coherent entities to get a higher chances to re-start a random walk, and thus their nodes will probably turn to get a higher steady state probability (i.e., relevance score) via the PageRank computation (Haveliwala, 2002). In this computation a significant role will be played by the weighted edges of the graph G_a which explicitly model the *semantic relatedness* among the entities mentioned by a.

The second attribute that is associated to each node is a *vector* of floating-point numbers computed through the DeepWalk model for entity embeddings (see Section 5.2). This technique is inspired by the approach adopted by Van Gysel et al. (2017), where the expertise of

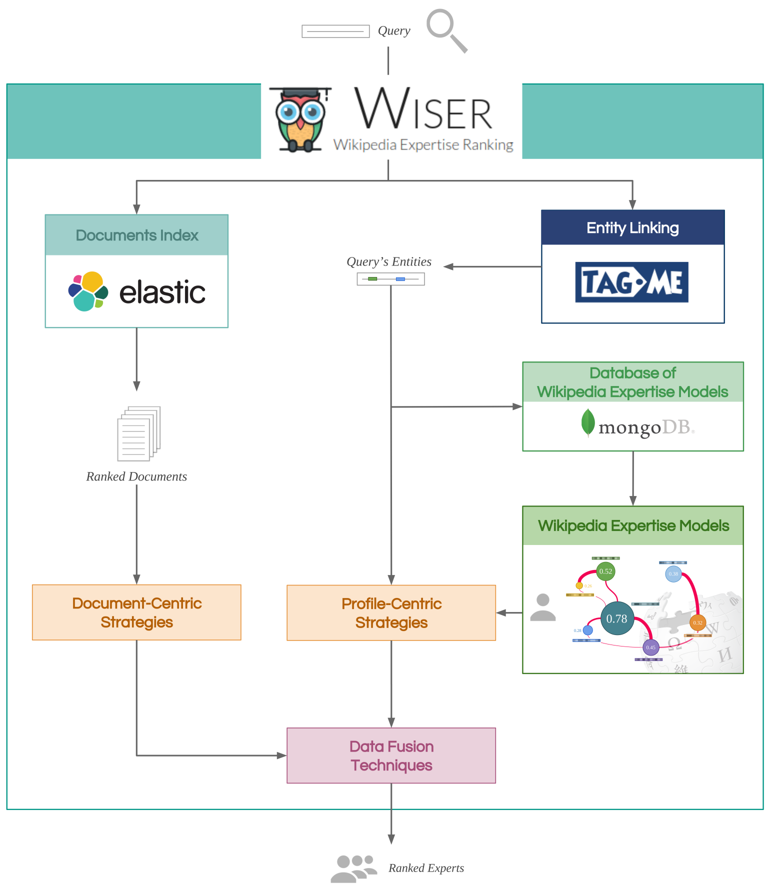

Query
WISER
Wikipedia Expertise Ranking
Documents Index
elastic
Entity Linking
TAGME
Query's Entities
Database of Wikipedia Expertise Models
mongoDB.
Ranked Documents
Wikipedia Expertise Models
0.78
Document-Centric Strategies
Profile-Centric Strategies
Data Fusion Techniques
Ranked Experts

Figure 5.2: Experts retrieval in WISER via the combination of a document-centric strategy and a profile-centric strategy through proper data fusion techniques that are described in the text.

each author is modeled with an embedding vector. But, unlike Van Gysel et al. (2017) where vectors are learned via a bag-of-words paradigm directly from the dataset (D, A), our embedding vectors are more "powerful" in that they embed the latent knowledge learned from the content and the structure of Wikipedia and, additionally, they "combine" the relevance score just described above and associated to each entity (node) in the graph G_a. Eventually, we compute for every author a one single embedding vector which is obtained by summing up the DeepWalk embedding vectors relative to its top-k entities and ranked according to the relevance score described above.* This embedding vector eventually incorporates the expertise

*In the experiments of Section 5.5 we will investigate the impact of the choice of $k \in \{10, 20, 30, 50, |E_a|\}$.

of each author into 100 components (see Section 5.5), thus it is fast to be managed in the subsequent query operations when we will need to compute the *semantic matches* between authors' topics and query topics.

Summarizing, WISER computes for every author a its WEM *profile* which consists of the graph G_a and an embedding vector of 100 numeric components. This way the WEM profile models the *explicit knowledge* of author a by identifying the explicit concepts (via entities and their relations) and the latent concepts (via an embedding vector) occurring in her documents. The graph G_a is crucial in many aspects because it captures the entities mentioned in a's documents and their Milne&Witten's relatedness score. But, also, it allows to select the top-k entities that best describe a's expertise, according to a relevance score derived by means of a PageRank calculation over G_a. The DeepWalk vectors of these top-k entities are then summed to get the embedding vector of author a that describes the best latent concepts of a's expertise.

5.4.2 Finding the Experts

At query time, WISER operates in order to identify the expertise areas mentioned in the input query q and then retrieve a set of candidate experts to which it assigns an expertise score. This score is eventually used for generating the final ranking of experts that are returned as result of query q.

Since our system relies on both *document-centric* and *profile-centric strategies*, we organized this section in three main paragraphs which respectively describe each of those strategies and the method used for their combination via proper data fusion techniques. Figure 5.2 reports a graphical representation of the *query* processing phase.

Document-Centric Strategy. It relies on the use of Elasticsearch. The query q is forwarded to Elasticsearch in order to retrieve a ranked list of documents, namely a list $(d_1, s_1), \ldots, (d_n, s_n)$ where s_i is the score computed for document d_i given the query q. In our experiments we will test several ranking scores: TF-IDF (Manning et al., 2008), BM25 (Robertson et al., 2009), and Language Modeling with either Dirichlet or Jelinek-Mercer smoothing ranking techniques (Zhai & Lafferty, 2017).

Table 5.1: Document-scoring techniques used by WISER within its document-centric strategy. We denote by $s_{a,j}$ the score assigned to the j-th document of author a computed via several techniques.

Name	**Equation**	**Description**
MEAN-k	$\frac{1}{k}\sum_{j=1}^{k} s_{a,j}$	Average of the top-k scores of a's documents.
MAX	$\max(s_{a,j})$	Maximum of the scores of a's documents.
RR	$\sum_{j=1}^{\lvert D_{a,q}\rvert} \frac{1}{\mathrm{rank}(d_j)}$	Reciprocal rank (Macdonald & Ounis, 2006) of the ranks of a's documents. $\mathrm{rank}(d_j)$ is the ranking position of document d_j.
COMBNZ	$\frac{\lvert D_{a,q}\rvert}{\lvert D_a\rvert} \sum_{j=1}^{\lvert D_{a,q}\rvert} s_{a,j}$	Documents' scores of author a, normalized by the number of documents associated to a.

The ranked list of documents is then turned into a ranked list of authors $a_1, \ldots, a_m$ by means of several well-known techniques (Fox & Shaw, 1994; Macdonald & Ounis, 2006) that we have adapted to our context, are described in Table 5.1 and tested in Section 5.5.

Profile-Centric Strategy. This is a novel set of scoring strategies that we have specifically designed to rank experts according with our new WEM profile. Authors are scored via a computation that consists of three main steps. First, WISER runs TAGME over the input query q and annotates it with a set of pertinent Wikipedia entities, denoted by E_q. Second, it retrieves as candidate experts the authors A^q whose WEM profile contains at least one of the entities in E_q. Third, the authors in A^q are ranked according to two novel *entity-scoring* methods, that we call *exact* and *related*, which compute authors' scores based on some properly defined *exact-* or *related-*scoring functions that are computed between q and their WEM profiles. These many scoring functions will be experimentally tested in Section 5.5.

Exact-Match Scoring. This collection of methods measures the relevance of an author $a \in A^q$ with respect to the query q as a function of the *frequency* of E_q's entities which occur in a's documents. More precisely, an author $a \in A^q$ is first retrieved as candidate expert of q if her WEM profile contains *at least one* of the entities annotated in E_q; and then, she is ranked by means of one of the techniques reported in Table 5.2 that take into account only the frequency of the entities explicitly occurring in its WEM profile.

Relate-Match Scoring. This approach aims at implementing a *semantic scoring* of the authors in A^q, by evaluating the pertinence of the expertise of an author $a \in A^q$ according to the *relatedness* among the entities in her WEM profile and the entities in E_q (as opposite to the

Table 5.2: Author-scoring techniques based on *exact-match* of entities and used by WISER within its profile-centric strategy. The function f can be linear, sigmoid or a square function. Equation EC-IAF, EF-IAF and REC-IAF are computed for a given author a and entity e, whereas MAX and MEAN aggregate these scores computed for multiple entities into a single one.

Name	Equation	Description
IAF	$\log \frac{\lvert A\rvert}{\lvert A_e\rvert}$	Inverse author frequency, namely the *smoothing factor* used for modeling the importance of entity e in the dataset at hand. This score is used only when combined with other techniques (see EC-IAF and EF-IAF).
EC-IAF	$\lvert D_{a,e}\rvert \cdot \rho_{a,e} \cdot \text{IAF}(e)$	Frequency of an entity smoothed by means of its coherence with a's documents (i.e., $\rho_{a,e}$) and the IAF scores.
EF-IAF	$\frac{1}{\lvert D_a\rvert} \cdot \text{EC-IAF}(a,e)$	Scaling down EC-IAF by means of the "productivity" of author a measured as the number $\lvert D_a\rvert$ of authored documents.
REC-IAF	$f(r_{a,e}) \cdot \text{EC-IAF}(a,e)$	Extending EC-IAF equation with the relevance score $r_{a,e}$ of the entity e within the graph G_a. $f(r_{a,e})$ is a scaling function described in the experiments.
MAX	$\max(g(a,e))$	Maximum exact-match score computed for a given author $a \in A^q$ and for each $e \in E_q$. $g(a,e)$ is either EC-IAF, EF-IAF or REC-IAF.
MEAN	$\text{mean}(g(a,e))$	Average exact-match score computed for a given author $a \in A^q$ and for each $e \in E_q$. $g(a,e)$ is either EC-IAF, EF-IAF or REC-IAF.

Table 5.3: Author-scoring techniques based on *related-match* of entities and used by WISER within its profile-centric strategy. The top-k entities of author a are the ones with the highest relevance score in G_a. In the experiment we have set $k = 0.1 \cdot \lvert A_e\rvert$, thus taking the top 10% entities mentioned in a's documents.

Name	Equation	Description
AER	$\frac{1}{k\,\lvert E_q\rvert}\sum_{e_q\in E_q}\sum_{i=1}^{k}\rho_{e_{a,i},a}\cdot \text{rel}(e_q, e_{a,i})$	Author entity relatedness score among the top-k entities of a and the entities $e_q \in E_q$.
RAER	$\frac{1}{k\,\lvert E_q\rvert}\sum_{e_q\in E_q}\sum_{i=1}^{k}\rho_{e_{a,i},a}\cdot \text{rel}(e_q, e_{a,i})\cdot f(r_{a,e_{a,i}})$	Ranked author entity relatedness score that extends AER with entities' relevance score. $f(r_{a,e})$ is a scaling function described in the experiments.
AES	$\text{cosine}(\sum_{e_q\in E_q}\vec{v}_{e_q} \cdot \vec{v}_{a,k})$	Author entity relatedness that computes the cosine similarity between the embedding $\vec{v}_{e_q}$ of entity $e_q \in E_q$ and the embedding $v_{a,k}$ of author a.

Table 5.4: Data fusion techniques used by WISER to combine h scores (document-centric and profile-centric) of an author a into a unique value that reflects the pertinence of a's expertise with the user query q.

Name	Equation	Description
COMBSUM	$\sum_{i=1}^{h} s_i(q,a)$	The final score is the sum of the scores.
COMBMIN	$\min_{i=1}^{h} s_i(q,a)$	The final score is the minimum of the scores.
COMBMAX	$\max_{i=1}^{h} s_i(q,a)$	The final score is the maximum between the scores.
RRM	$\prod_{i=1}^{h} \frac{1}{\text{rank}_i(q,a)}$	The final score is the product of the inversed ranking scores.
RRS	$\frac{1}{\sum_{i=1}^{h} \text{rank}_i(q,a)}$	The final score is the inverse of the sums of the ranking scores.

frequency used by the previous scoring functions). Table 5.3 reports the list of techniques used to design such a kind of *semantic scores*. They exploit either the structure of the graph G_a (i.e., AER and RAER) or compute the cosine similarity between the embedding vectors of the compared entities (i.e., AES).

Combining Document-Centric and Profile-Centric Strategies. Document and profile-centric strategies are then eventually combined via proper data fusion techniques which are listed in Table 5.4. We designed those techniques as adaptations of the proposals in (Fox & Shaw, 1994; Macdonald & Ounis, 2006) suitable for the experts finding problem.

5.4.3 Optimization and Efficiency Details

WISER implements three main algorithmic techniques that speed-up the retrieval of experts, thus making the query experience user-friendly.

Double Index. WISER's index is implemented with two different data structures, namely, two inverted lists that store both the association author-entities and entity-authors. This allows to efficiently retrieve at query time all the information that are needed for ranking authors with profile-centric strategies.

Ordered Entities by Relevance Score. Some profile-centric strategies, namely AER and RAER, need to retrieve the top-k most related entities of an author with respect to E_q, but this latter set of entities is known only at query time. This could be a slow process when dealing with many authors and many entities, so WISER pre-computes and stores for each author the ordered list of her entities sorted by their relevance score, computed by means of a PageRank over G_a (i.e., a's WEM profile). The computation of the top-k entities in E_a with respect to E_q then boils down to a fast computation of a list intersection.

Relatedness Cache. The indexing phase of WISER needs to compute the graph G_a for every author a of the input dataset. This could be a very slow process in the presence of many authors a and many entities in E_a, because G_a is a graph of up to $\Theta(|E_a|^2)$ edges which have to be weighted by querying the RESTful service underlying the WikipediaRelatedness framework. In order to speed up this computation, WISER caches the edge weights as soon as they are computed. This way, if two entities occur in many subsequent graphs G_a, their computation is saved by accessing their cached values.

5.5 Experiments

In order to evaluate the efficacy of WISER we have set up a sophisticated experimental framework that has systematically tested the various document-centric and profile-centric strategies described in Tables 5.1–5.3 and the data fusion techniques described in Tables 5.4 over the publicly available TU dataset (Berendsen et al., 2013). From these experiments we will derive the best combination of techniques that, then, will be used to compare the resulting WISER against the state-of-the-art systems currently known to solve the expert finding problem.

5.5.1 Dataset

The TU (Berendsen et al., 2013) dataset[*] is an updated version of the UvT dataset, developed at Tilburg University (TU). It is currently the largest dataset available for benchmarking academia expert finding solutions, containing both Dutch and English documents. TU dataset comes with five different (human assessed) ground-truths, named from GT1 to GT5. In our experiments we have decided to use GT5 because it is considered the most recent and complete ground-truth (see (Berendsen et al., 2013) for details) and because it is the dataset used in the experiments of (Van Gysel et al., 2016b). Table 5.5 offers a high-level overview about the dataset, while Table 5.6 offer a finer description.

[*]We thank Christophe Van Gysel for providing us the dataset.

Table 5.5: Overview of the TU dataset (Van Gysel et al., 2016b).

Resource	Count
Documents	31209
Author Candidates	977
Queries (GT5)	1266
Document-candidate associations	36566
Documents with at least one associated candidate	27834
Associations per document	1.13 ± 0.39
Associations per candidate	37.43 ± 61.00

Table 5.6: Document composition for the TU dataset.

Resource	Documents with at least one author	Documents with no authors	Total num. documents
Theses	5152	871	6023
Papers	21120	2504	23624
Profile pages (UK)	495	0	495
Profile pages (NL)	524	0	524
Course pages	543	0	543
Total documents	27834	31209	3375

Table 5.7: Space occupancy of WISER's index built on the TU dataset.

Resource	Space
Raw Documents	25 MB
Elasitcsearch Index	40 MB
WEM Profiles (total)	94 MB
WEM Profiles (average per author)	100 KB

Indexing TU with Wiser. Since TU dataset contains both Dutch and English documents, we *normalize* the data collection by translating Dutch documents into English via the tool Translate Shell[*]. Then, the dataset is indexed with Wiser, as described in Section 5.4.1. Table 5.7 reports the memory occupancy of the final indexes.

5.5.2 Evaluation Metrics

In our experiments we will use the following ranking metrics that are available in the trec_eval tool[†], and are commonly used to evaluate expert-finding systems.

Precision at k (P@k). It is the fraction of retrieved authors that are relevant for a given query q with respect to a given cut-off k which considers only the topmost k results returned by the evaluated system:

$$\text{P@}k(q) = \frac{|\{\text{relevant authors for } q\} \cap \{\text{ top-}k \text{ retrieved authors for } q\}|}{k} \tag{5.1}$$

Mean Average Precision (MAP). Precision and recall are set-based measures, thus they are computed on unordered lists of authors. For systems that return ranked results, as the ones solving the expert-finding problem, it is desirable to consider the order in which the authors are returned. The following score computes the average of P@k over the relevant retrieved authors.

$$\text{AveP}(q) = \frac{\sum_{k=1}^{n} \text{P@}k(q) \times \text{rel}_q(k)}{|\{\text{relevant authors for } q\}|} \tag{5.2}$$

where n is the number of retrieved authors, $\text{rel}_q(k)$ is function which equals to 1 if the item at rank k is a relevant author for q, 0 otherwise.

The following score averages AveP over all queries in Q.

$$\text{MAP} = \frac{\sum_{q \in Q} \text{AveP}(q)}{|Q|} \tag{5.3}$$

[*]An open source command-line translator via Google Translate APIs.

[†]github.com/usnistgov/trec_eval

Mean Reciprocal Rank (MRR). The reciprocal rank of a query response is the inverse of the rank of the first correct answer for q (i.e., $\text{rec-rank}(q)$), namely:

$$\text{rec-rank}(q) = \frac{1}{\text{pos}(q)} \tag{5.4}$$

The following score averages the reciprocal rank over all queries in Q:

$$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \text{rec-rank}(q) \tag{5.5}$$

Normalized Discounted Cumulative Gain (NDCG). Assuming to have a relevance score for each author, given a query q, we wish to have measures that give more value to the relevant authors that appear high in the ranked list of results returned for q. Discounted Cumulative Gain is a measure that penalizes highly relevant authors appearing lower in the result list for q. This is obtained by reducing their relevance value (i.e., rel_q, see above) by the logarithmic of their position in that list.

$$\text{DCG}_k(q) = \text{rel}_\text{q}(1) + \sum_{i=2}^{k} \frac{\text{rel}_\text{q}(i)}{\log_2 i} \tag{5.6}$$

The final measure we introduce for our evaluation purposes is among the most famous ones adopted for classic search engines (Manning et al., 2008). It is computed by normalizing DCG with respect to the best possible ranking for a given query q. More precisely, for a position k, the Ideal Discounted Cumulative Gain — $\text{IDCG}_k(q)$ — is obtained by computing the $\text{DCG}_k(q)$ on the list of authors sorted by their relevance score wrt q. Then the measure $\text{NDCG}_k(q)$ is obtained as the ratio between $\text{DCG}_k(q)$ and $\text{IDCG}_k(q)$:

$$\text{NDCG}_k(q) = \frac{\text{DCG}_k(q)}{\text{IDCG}_k(q)} \tag{5.7}$$

5.5.3 Results and Analysis

Section 5.4 has described several possible techniques that Wiser can use to implement its document-centric and profile-centric strategies. In this section we experiment all these pro-

Table 5.8: Comparison among different configurations of document-centric strategies with normalized reciprocal rank (RR) as data-fusion technique.

Method	**MAP**	**MRR**	**P@5**	**P@10**	**NDCG@100**
TF-IDF (RR)	0.284	0.347	0.120	0.082	0.420
BM25 (RR)	**0.363**	**0.437**	**0.157**	**0.099**	**0.495**
LM (Dirichlet, RR)	0.341	0.410	0.145	0.096	0.473
LM (Jelinek-Mercer, RR)	0.346	0.414	0.151	0.098	0.481

posals by varying also their involved parameters. More precisely, for the document-centric strategies we experiment different document rankings and investigate also several data-fusion techniques that allow us to assign one single score to each candidate expert given all of its documents that are pertinent with the input query (see Tables 5.1 and 5.4). For the profile-centric strategies, we experiment the *exact-* and *related-*match scoring methods summarized in Tables 5.2 and 5.3. At the end, from all these figures we derive the best possible configurations of WISER, and then compare them against the state-of-the-art approaches (Van Gysel et al., 2016b). This comparison will allow us to eventually design and implement a state-of-the-art version of WISER that further improves the best known results, by means of a proper orchestration of document-centric, profile-centric and data-fusion strategies. Finally, in the last part of this section we will conclude the experiments with a run-time evaluation and a qualitative analysis that will show how the combination of document- and profile-centric strategies does not only improve the quality of the returned results, but it also does not significantly alter the latency response of the system.

Evaluation of the Document-Centric Strategies. We configure WISER to first rank documents via various scoring functions: i.e. TF-IDF, BM25, or Language Modeling with Dirichlet or Jelinek-Mercer smoothing. Then, we compute a score for each author that combines two or more of the previous rankings via one of the data-fusion techniques described in Section 5.4.2 and summarized in Table 5.4. As far as the smoothing configurations for Dirichlet or Jelinek-Mercer approaches are concerned, we set $\mu = 2000$ and $\lambda = 0.1$, as suggested by the documentation of Elasticsearch.

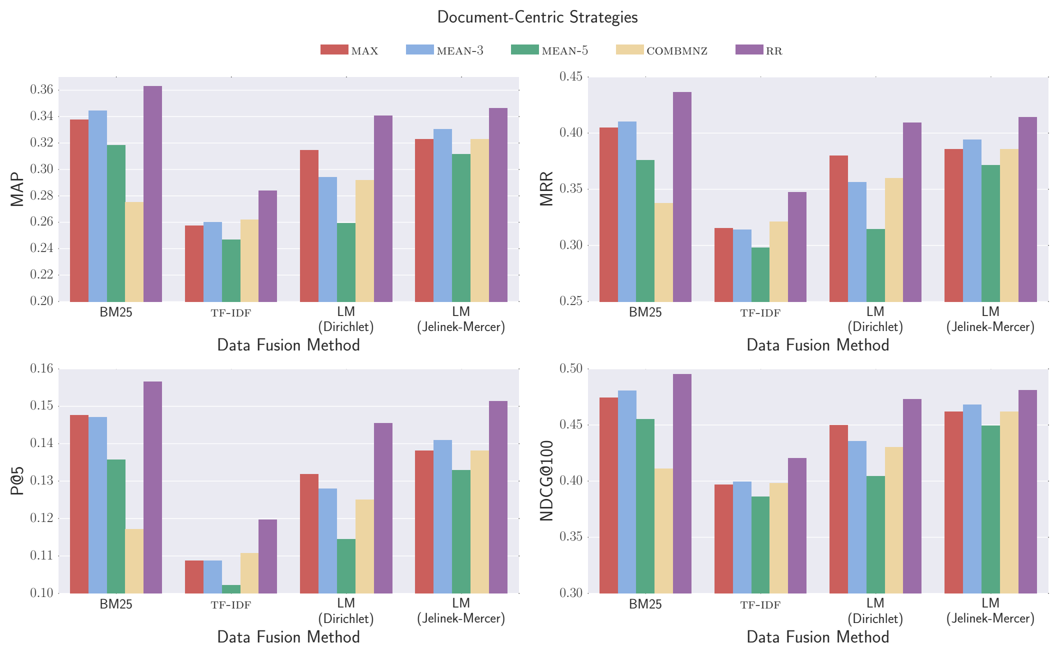


Figure 5.3: Expert finding performance of WISER with different configurations of document-centric strategies and data-fusion methods.

Figure 5.3 reports the performance of WISER by varying: (1) the document ranking, (2) the data fusion method, and (3) the evaluation metric. Looking at the histograms, it is very clear that each strategy achieves the best performance when the reciprocal rank (rr in the Figures) is used as data-fusion method. So, we have set rr in our following experiments and explored the best performance for all other combinations. Results are reported in Table 5.8 below. We notice that, despite all strategies have values of P@5 and P@10 very close to each other, a difference is present on MAP, MAP and NDCG@100. As far as the document-rankings are concerned we note that TF-IDF is the worst approach, whereas both LM strategies have good performance and, undoubtly, BM25 is the clear winner with +7.9% on MAP, +9% on MAP and +7.5% on NDCG@100 with respect to TF-IDF, and +1.7% on MAP and +2.3% on MAP and +1.4% on NDCG@100 with respect to any LM. So the winner among the document-centric strategies is BM25 with RR as data-fusion method.

Evaluation of the Profile-Centric Strategies. We experimented the two configurations of WISER that deploy either the *exact-* or the *related*-match score for evaluating the pertinence of the WEM profile of an author with respect to the entities of a given query, as described in

Section 5.4.2. To ease the reading of the following experimental results, we will first comment on their individual use and then illustrate some combinations.

Exact-Match Scoring. Figure 5.4 reports the performance of Wiser configured to rank authors either with ec-iaf or ef-iaf (both methods based on entity frequency) and by deploying max and mean methods for combining multiple scores into a single one. It is evident that ec-iaf scoring with mean outperforms ef-iaf.

Figure 5.5 shows the performance of Wiser with different configurations of rec-iaf scoring, PageRank executed over the author's graph G_a). Since rec-iaf depends on $f(r_{a,e})$, we experimented various settings for f that we report on the top of Figure 5.5, namely identity function (linear), sigmoid function (sigmoid), square root function (sqrt), and square function square. Looking at the plots, it is evident that the best configuration for rec-iaf is achieved when f is the square function, it improves both ec-iaf or ef-iaf.

Related-Match Scoring. Figure 5.6 shows the performance of aer and raer profile-centric strategies. Since raer depends on $f(r_{a,e})$, we have investigated the same set of scaling functions experimented for the rec-iaf method. Despite the fact that the raer method works slightly better when configured with the sigmoid function, the simpler aer method is equivalent or slightly better on all metrics.

Figure 5.7 reports the performance of Wiser which ranks authors according to DeepWalk embeddings models, which have been learned via cbow algorithm and by fixing the size of the vectors to 100. In those experiments we have also evaluated the impact of varying the number k of top-k entities selected per author. As the plots show, ranking experts with respect to the DeepWalk embedding achieves better performance on different metrics and is more robust with respect to the k parameter. In the following experiments we have set $k = 30$. For the sake of completeness, we mention that we have also investigated the application of DeepWalk Skip-gram and Entity2Vec (Ni et al., 2016) (both cbow and Skip-gram) models, but for the ease of explanation we did not report them since their performance are lower than DeepWalk-cbow.

Final Discussion. Table 5.9 reports the best configuration found for each profile-centric method, as derived from the previous Figures. Generally speaking, methods based on *exact*-match perform better than the ones based on *related*-match on the TU dataset, with rec-iaf that achieves

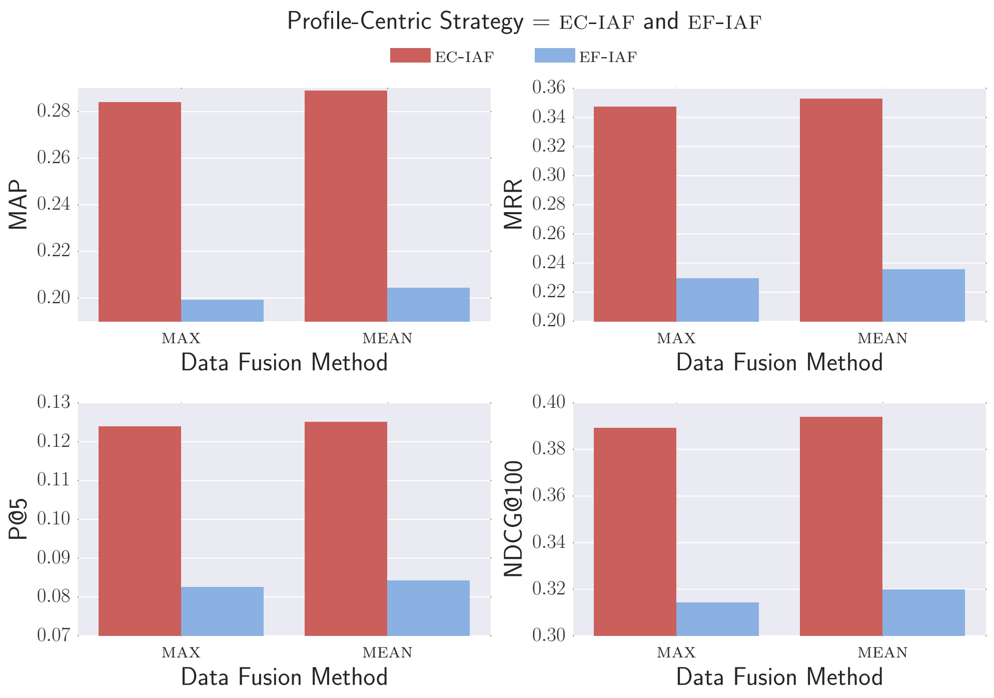


Figure 5.4: Performance of WISER by profile-centric strategies based on entity count: ec-iaf and ef-iaf.

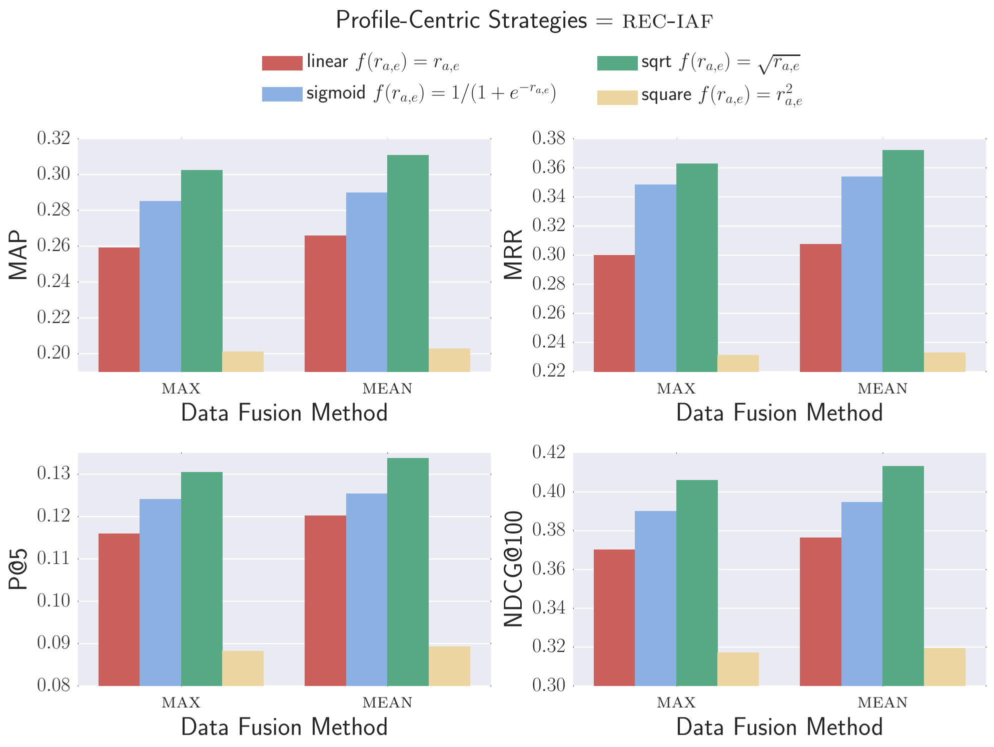


Figure 5.5: . Performance of WISER by rec-iaf as profile-centric strategy with different scaling functions $f(r_{a,e})$ for the relevance score $r_{a,e}$.

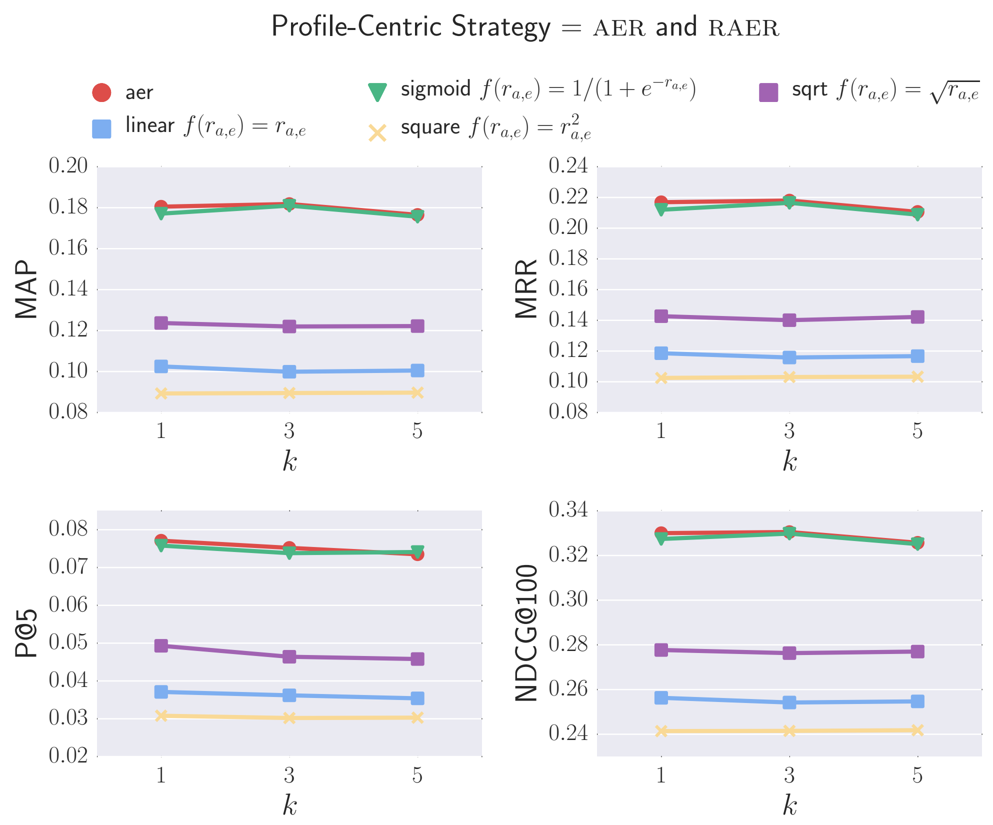


Figure 5.6: . Expert finding performance of WISER by deploying aer and different configurations of raer.

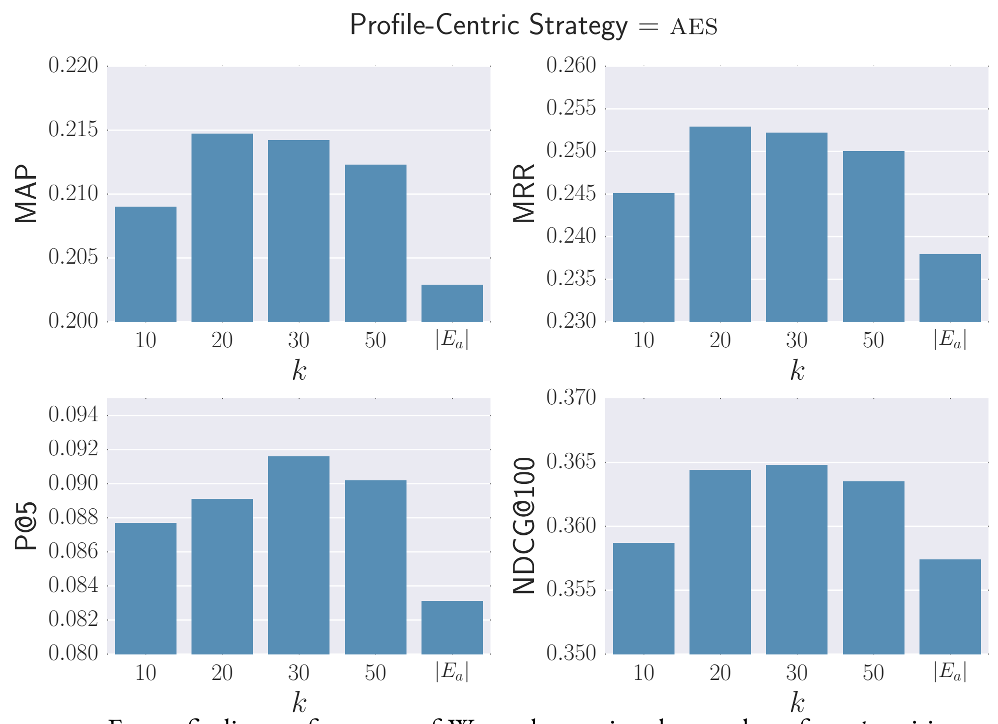


Figure 5.7: . Expert finding performance of WISER by varying the number of top-k entities used for generating the unique embedding vector of each author with DeepWalk-CBOW model.

Table 5.9: Comparison between the best configuration of the profile-centric approaches (with both exact and related match scoring methods) implemented by WISER.

Match	Method	MAP	MRR	P@5	P@10	NDCG@100
Exact	EC-IAF (MEAN)	0.289	0.353	0.125	0.081	0.394
	EF-IAF (MEAN)	0.204	0.236	0.084	0.064	0.320
	REC-IAF (SQRT-MEAN)	**0.311**	**0.372**	**0.134**	**0.086**	**0.413**
Related	AER	0.187	0.226	0.081	0.058	0.332
	RAER (SIGMOID)	0.185	0.224	0.081	0.058	0.331
	AES (DW-CBOW-30)	0.214	0.255	0.092	0.067	0.365

a peak of +9.7% on MAP with respect to the aes method. It is crucial to stress at this point the role of the WEM profile of an author a in achieving these results. In fact, the best methods — i.e. for the *exact*-match (i.e., REC-IAF), the *related*-match (i.e., AES) — are properly the ones that strongly deploy the weighted graph G_a to derive, via a PageRank computation, the relevance scores $r_{a,e}$ for the entities e mentioned within a's documents and the corresponding top-k entities.

WISER versus the State-of-the-Art. In this last paragraph we compare the best configurations of WISER, based on document- and profile-centric methods, against the best known approaches present in literature, i.e. Log-liner (Van Gysel et al., 2016b) and Model 2 (JM) (Balog et al., 2006).

Table 5.10 shows that both BM25 and REC-IAF methods outperform Log-linear and Model 2 (JM) over different metrics. Specifically, REC-IAF achieves competitive performance with an improvement of +2.4% over the MAP and +0.9% over MAP scores with respect to Log-linear, whereas BM25 improves all knwon methods over all metrics: +7.6% on MAP, +7.4% on MAP, +2.3% on P@5, +0.7% on P@10 and +7% on NDCG@100, thus resulting the clear winner and showing that for the TU dataset the document-centric strategy is better than the profile-centric strategy in WISER.

Given these numbers, we set up a final experiment that aimed at evaluating the best performance achievable by the combination of these methods via data-fusion techniques. Specifically, we designed a version of WISER that combines the best document-centric strategy, i.e. BM25 (RR), with the two best profile-centric strategies, i.e. REC-IAF and AES. Figures 5.8 and

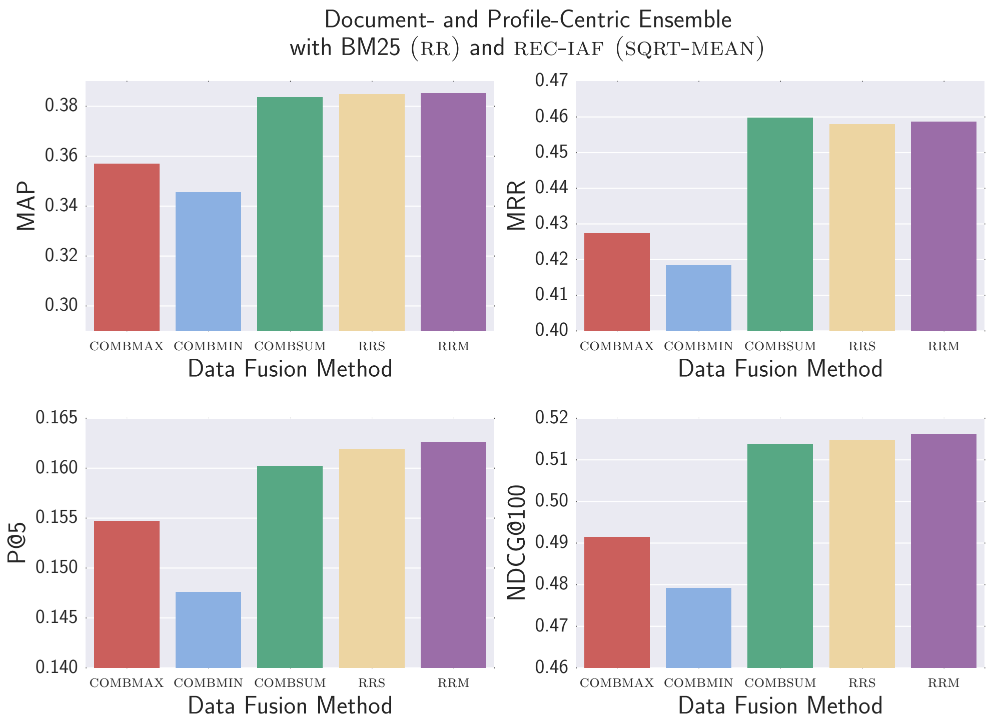


Figure 5.8: . Performance of WISER configured to combine document-centric (i.e., BM25 (RR)) and a profile-centric strategy — i.e., rec-iaf (sqrt-mean) — by means of several data-fusion techniques.

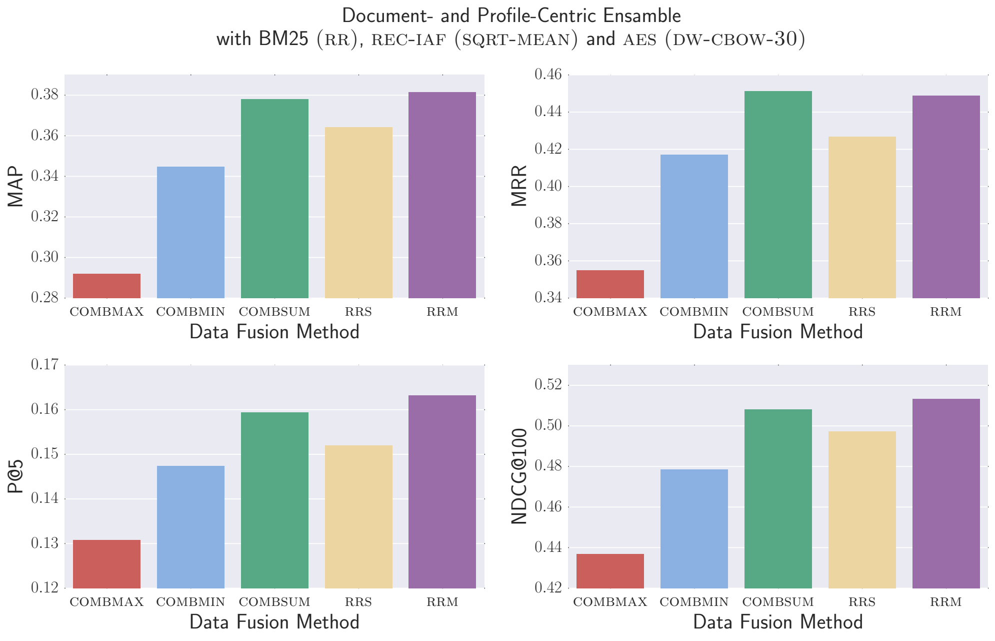


Figure 5.9: . Performance of WISER configured to combine document-centric — i.e., BM25 (RR) — the best exact profile-centric strategy — i.e., REC-IAF (SQRT-MEAN) — and the best related profile-centric strategy — i.e., AES (DW-CBOW-30) — by means of several data-fusion techniques.

Table 5.10: Comparison between the best approaches reported in literature (top) and WISER's variants (bottom). Statistical significance of BM25 (RR) is computed using a two-tailed paired t-test with respect to REC-IAF (SQRT-MEAN) and indicated with $^\blacktriangle$ when $p < 0.01$.

Method	MAP	MRR	P@5	P@10	NDCG@100
Model 2 (JM) (Balog et al., 2006)	0.253	0.302	0.108	0.081	0.394
Log-linear (Van Gysel et al., 2016b)	0.287	0.363	0.134	0.092	0.425
BM25 (RR)	**0.363**$^\blacktriangle$	**0.437**$^\blacktriangle$	**0.157**$^\blacktriangle$	**0.099**$^\blacktriangle$	**0.495**$^\blacktriangle$
REC-IAF (SQRT-MEAN)	0.311	0.372	0.134	0.086	0.413
AES (DW-CBOW-30)	0.214	0.255	0.092	0.067	0.365

Table 5.11: Comparison between single methods (top) and different ensemble techniques whose ranking are combined via RRM data-fusion method. Statistical significance is computed using a one-tailed paired t-test with respect to BM25 (RR) — the best method of Table 5.10 — and indicated with $^\triangle$ for $p < 0.1$) and $^\blacktriangle$ for $p < 0.05$.

Method	MAP	MRR	P@5	P@10	NDCG@100
Model 2 (JM) (Balog et al., 2006)	0.253	0.302	0.108	0.081	0.394
Log-linear (Van Gysel et al., 2016b)	0.287	0.363	0.134	0.092	0.425
BM25 (RR)	0.363	0.437	0.157	0.099	0.495
REC-IAF (SQRT-MEAN)	0.311	0.372	0.134	0.086	0.413
AES (DW-CBOW-30)	0.214	0.255	0.092	0.067	0.365
Ensemble (Van Gysel et al., 2016b)	0.331	0.402	0.156	**0.105**	0.477
RRM(BM25 (RR), REC-IAF (SQRT-MEAN))	**0.385**$^\triangle$	**0.459**$^\triangle$	**0.163**	0.104	**0.516**$^\blacktriangle$
RRM(BM25 (RR), REC-IAF (SQRT-MEAN), AES (DW-CBOW-30))	0.381$^\triangle$	0.449	**0.163**	**0.105**$^\triangle$	0.513$^\triangle$

5.9 report the performance of these combinations. The best performance are always reached when the methods at hands are combined with the RRM data-fusion method (purple bar).

Table 5.11 reports the performance achieved by the best known and new approaches proposed in this chapter. For the sake of comparison, we also report the Ensemble method developed by (Van Gysel et al., 2016b), which combines via reciprocal rank (i.e., RR) the Log-linear model with Model 2 (JM). It is evident from the table that

- the BM25 (RR) implemented by WISER outperforms the Ensemble method of (Van Gysel et al., 2016b), which is currently the state-of-the-art, of +3.2%, +3.5% and 1.8% in MAP, MRR and NDCG@100, and

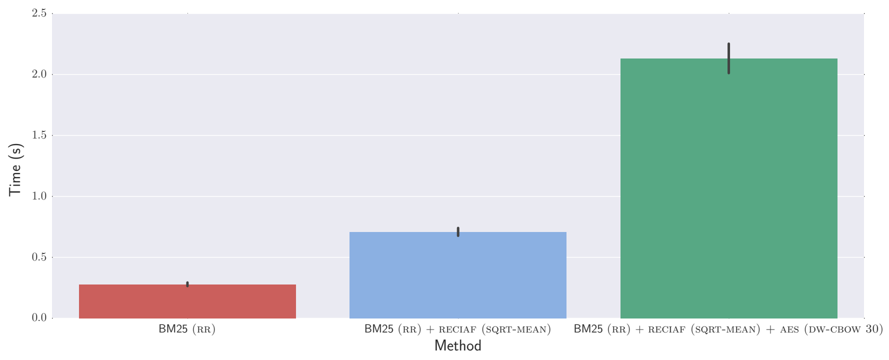


Figure 5.10: Average tun-time performance of the three best configurations of WISER on the TU dataset.

- with a proper combination of this document-centric strategy with the two best profile-centric algorithms of WISER we are able to achieve a further improvement over Ensemble on MAP, MAP and NDCG@100 of +5.4%, +5.7%, +0.7% and +3.9%.

Therefore, WISER turns out to be the new state-of-the-art solution for the expert finding problem in the academia domain.

Run-Time Evaluation. We supplement the large-scale quantitative evaluation described in the previous paragraph with a run-time evaluation performed on the top of the three best configurations of WISER. All tests that we report here were performed on an Intel Core i7-4790 clocked at 3.60GHz, with 16GB of RAM and running Linux 4.13.

Unfortunately, we can not compare the speed of WISER against to the other known systems — i.e. Model 2 (JM) (Balog et al., 2006), Log-linear (Van Gysel et al., 2016b) and Ensemble (Van Gysel et al., 2016b) — because they are not publicly available.

As far as the time efficiency of the three best configurations of WISER is concerned — i.e., BM25 (RR), RRM (BM25 (RR), rec-iaf (sqrt-mean)) and RRM (BM25 (RR), REC-IAF (SQRT-MEAN), AES (DW-CBOW-30)), our experiments on the TU dataset show that WISER is able to increase significantly its output quality but at a time cost which results not negligible for its third tested configuration. As expected, Figure 5.10 shows that the simplest scoring strategy — i.e., BM25 (RR) — is the fastest one, since it needs only to retrieve the relevant documents for a given query and then compute the rr score for each candidate author. On the other hand, the second configuration RRM (BM25 (RR), REC-IAF (SQRT-MEAN)) improves the quality of

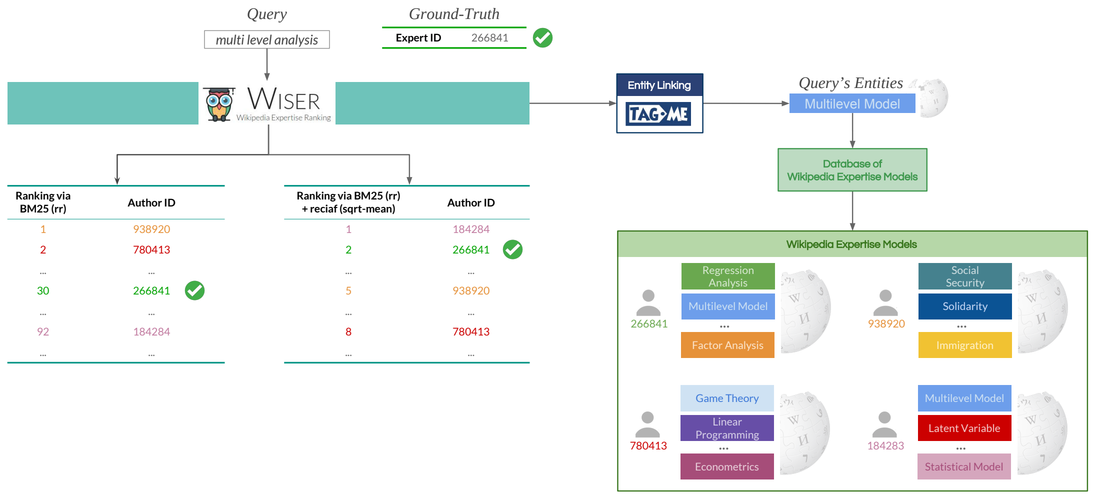


Figure 5.11: A real example that shows where the profile-centric approaches complement efficaciously the document-centric approaches. On the top, the input query and the ground-truth expert. On the left the final results returned by WISER with two different configurations. On the right, a simplified internal architecture of the system that shows the entity annotated in the input query and a meaningful subset of entities for each ranked experts in order to show the topics concerned by their research.

the results returned by BM25 (RR) (see Table 5.11) at a small time penalty, which makes the overall system still able to answer queries in less than one second. The third and last technique — i.e., RRM (BM25 (RR), REC-IAF (SQRT-MEAN), AES (DW-CBOW-30)) — incurs in a larger running time because it is more computationally intensive given that it needs to compute several cosine similarities between the entities annotated in the input query and the top-30 entities in the authors' profiles.

Accordingly with Table 5.11 and with the run-time performance of the three tested methods in Figure 5.10, we decided to deploy as the final configuration of WISER at wiser.d4science.org the one which has shown both the highest qualitative and the fastest run-time performance: namely, RRM (BM25 (RR), REC-IAF (SQRT-MEAN)).

Qualitative Analysis. In this last paragraph we wish to shed light on the combination between our new profile-centric approaches with the classic document-centric approaches. To fulfill this goal we have performed a qualitative analysis that consisted in manually inspecting the results returned by two configurations of WISER: one instantiated with a purely document-centric approach — i.e., BM25 (RR) — and the other one exploiting a combination between document- and profile-centric approaches, namely RRM (BM25 (RR), REC-IAF (SQRT-MEAN)). We have actually identified a common pattern that recurs very frequently when the results of BM25 (RR) are improved by RRM (BM25 (RR), REC-IAF (SQRT-MEAN)). Figure 5.11 shows one example of this common pattern for the query "multi level analysis". For this query BM25 (RR) ranks the best author (indicated in the ground truth as the one with ID equal to 266 841) at 20th position, and puts on the topmost positions many authors whose research is unrelated to the submitted query. This worse result is due to the fact that the bag-of-words paradigm underling BM25 incurs into the error of retrieving as experts those authors that have in their abstracts terms appearing in the query q but whose meaning is different from the one intended by q. In fact, in the example, the words *multi* and *analysis* appear frequently in the papers authored by 938 920 and 780 413, but with a meaning which is not related to the concept Multilevel Model intended by the query. Conversely, the profile-centric score based on Wikipedia entities and adopted by WISER allows to overcome this limitation. As we see in the right of Figure 5.11, TAGME correctly identifies the entity `Multilevel_Model` both in the query q and in the profile of the ground-truth expert (i.e., 266 841). On the other hand, TAGME does not annotate the profiles of the two non-experts, previously ranked in the top-2 positions by BM25 (RR) (i.e., 938 920 and 780 413), with that concept. As a result, the combination of document and profile-centric approaches implemented by RRM (BM25 (RR), REC-IAF (SQRT-MEAN)) allows to re-rank experts by scoring higher the ones whose profile contains the same entity of the input query q (i.e., 266 841) and, vice versa, demote the ones that include the query's terms but not their corresponding concepts (i.e., 938 920 and 780 413).

6
Conclusion and Future Directions

In this chapter we point out some future research directions that we believe are interesting and promising and concern, directly or indirectly, with the topics addressed in this thesis.

Entity Relatedness. The work presented in Chapter 3 leaves several open issues. We would like to apply our two-stage framework to other KGs and extend our approach by properly taking into account the *labels* that can be associated to entities' relationships. For example, in a KG like Wikidata and Yago, the edge that link between `Leonardo_da_Vinci` and `Anchiano` (the town where he was born) is labeled with the a string that specify this relationships, namely `place_of_birth`.

As a proof of concept, we have recently set-up one more experiment that investigates the application of our two-stage framework (configured with Step 1 and 3 with Milne & Witten (2008) relatedness, as done in Section 3.6.10) on the domain of *synonym extraction*: the task on which CoSimRank (Rothe & Schütze, 2014) has been originally designed for and experimented. For a fair comparison, we run our framework on the same graph of words deployed by Rothe & Schütze (2014), which consists of 30945 nodes and 2045411 edges, and over the same benchmark, namely TS68, a dataset published by Minkov & Cohen (2012) 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. The preliminary results of these experiments are reported in Table 6.1.

Table 6.1: Preliminary 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 (Rothe & Schütze, 2014).

Method	**P@1**	**MRR**	**Time**
PPR+Cos	20.6	32	$\mathcal{O}(mn^2)$
SimRank	25.0	**37**	$\mathcal{O}(n^3)$
CoSimRank	25.5	**37**	$\mathcal{O}(mn^2)$
Typed CoSimRank	23.5	**37**	$\mathcal{O}(mn^2)$
Two-Stage Framework	**26.6**	33	$\boldsymbol{\mathcal{O}(kdmn)}$

Our framework achieves competitive results, with performance 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 graph of words. These weights are actually exploited by all other methods reported in Table 6.1, thus further penalizing our results, especially in the MRR metric. This corroborate what we stated at the beginning of this paragraph: we clearly need to extend our framework incorporating the information present in the graph's edges (e.g., labels or weights) that are currently ignored during the relatedness computation of our approach.

Furthermore, a wider evaluation of relatedness methods is needed for studying the dependence of the performance from the link-richness of Wikipedia as well as assessing them over domain-specific entity categories (e.g., biology, medicine or finance). Our two-stage framework could also have an impact on other applications which hinge on the computation of large amounts of entity relatedness, such as query understanding (Cornolti et al., 2016), document relatedness (Ni et al., 2016), question answering (Abujabal et al., 2018) or, more generally speaking, to all that research contexts that need of a fast and effective relatedness computation between nodes of a graph.

Entity and Fact Salience. Our studies of entity and fact salience presented in Chapter 4 have highlighted several research issues and directions that need further investigation.

Improving the quality of the entity salience annotations of the NYT is a crucial need: augmenting its annotations with common nouns and labeling its ground-truth via crowdsourcing should improve the quality of the training set and thus of the trained systems as well as enable a larger and fairer assessment of their efficacy. Furthermore, we can also foresee that in the next years research will start focusing on the *efficiency* of entity linkers in order to boost the balancing between speed and accuracy that is becoming an emerging need for a knowledge extraction at large scale. For example, the current annotation of NYT by Wat, although it run on a multi-threaded machine, took about 20 days.

A number of applications in a myriad of domains are clearly possible, with the extraction of salient entities and salient facts that can be used by machines to enhance their understanding of documents by means of this novel salient-based representation. News credibility (Kashyap et al., 2018), knowledge base construction (Nguyen et al., 2017a) and facts contextualization (Voskarides et al., 2018) are only few examples of applications that could benefit from the extraction of salient entities and salient facts from an input document.

Expert Finding. The system Wiser introduced in Chapter 5 leaves several open issues that we would like to explore. For example, we need to design a fine-grain clustering technique for wem profiles in order to easily show which are the main *groups* of topics researched by the expert at the hand. For example, we found that clustering entities of a Computer Science researcher is not an easy task since all of them lie into the same domain. Here, classical approaches (e.g., k-means/hdbscan executed over wem) incur in the problem of generating coarse-grain clusters that group together almost all entities, despite they actually belong to different subareas. For example, over a research profile whose entities cover both the areas of algorithms (e.g., `Suffix_tree` and `Dynamic_programming`) and data compression (e.g., `Gzip` and `Burrows-Wheeler_transform`), a standard k-means generates just one single cluster. Looking closer to the results, we found that these coarse-grain clusters are generated because classical relatedness measures (such as Milne&Witten and entity embeddings) output higher scores for all of them and thus making almost impossible to distinguish between close and far entities. On the other hand, the application of our two-stage framework seems to preserve the original granularity and could enable the clustering algorithm to provide better groups of entities. This preliminary investigation can be explored in the profiles indexed at wiser.d4science.org, under the tab Main Areas.

The current on-line version of Wiser is limited to the data provided by the University of Pisa, which clearly needs to be extended with more resources and information for consequently increasing its coverage to other universities, institutions and countries.

Our new profiling technique wem has been experimented only on the domain of expert finding, while it is clearly generalizable and it could be applied to other research areas, such as social media and system recommendation.

A
GUI and Public API of Swat

This appendix describes the Graphical User Interface (GUI) and the public API our system Swat, which has been presented in Chapter 4.

Figure A.1 shows a simple GUI[*] that allows using Swat over an input document loaded via a Web interface. In addition to the GUI, it is possible to deploy Swat through a REST-like interface[†]. The API provides results in both human and machine-readable form, by deploying a simple JSON format (see Tables A.1, A.2 and A.3). In order to show how the interaction with Swat works, we offer a Python code snippet in Listing A.1 for querying our system and the corresponding JSON response in Listing A.2. A query requires just one optional parameter (i.e., title) and one mandatory parameter (i.e., the content of the document). The response includes all entities annotated by Swat and several information for each of them.

[*]The demo of the system is accessible at swat.d4science.org.
[†]The API is accessible at sobigdata.d4science.org/web/tagme/swat-api.

Listing A.1: Python code for querying the SWAT's public API. The authorization token `MY_GCUBE_TOKEN` is needed for using the service and obtainable through free registration.

```python
import json
import requests

MY_GCUBE_TOKEN = 'copy your gcube-token here!'

document = {
  "title": 'Obama travels.',
  "content": 'Barack Obama was in Pisa for a flying visit.'
}

response = requests.post('https://swat.d4science.org/salience',
                         data=json.dumps(document),
                         params={'gcube-token': MY_GCUBE_TOKEN})

print json.dumps(response.json(), indent=4)
```

Listing A.2: Structure of the JSON response of SWAT.

```
{
  'status'                      # str
  'annotations':
    {
      'wiki_id'                 # int
      'wiki_title'              # str
      'salience_class'          # int
      'salience_score'          # float
      'spans':                  # where the entity is mentioned in content
      [
        {
          'start'               # int (character-offset, included)
          'end'                 # int (character-offset, not included)
        }
      ]
  }
  'title'                       # str
  'content'                     # str
}
```

Table A.1: Fields of the SWAT's JSON request.

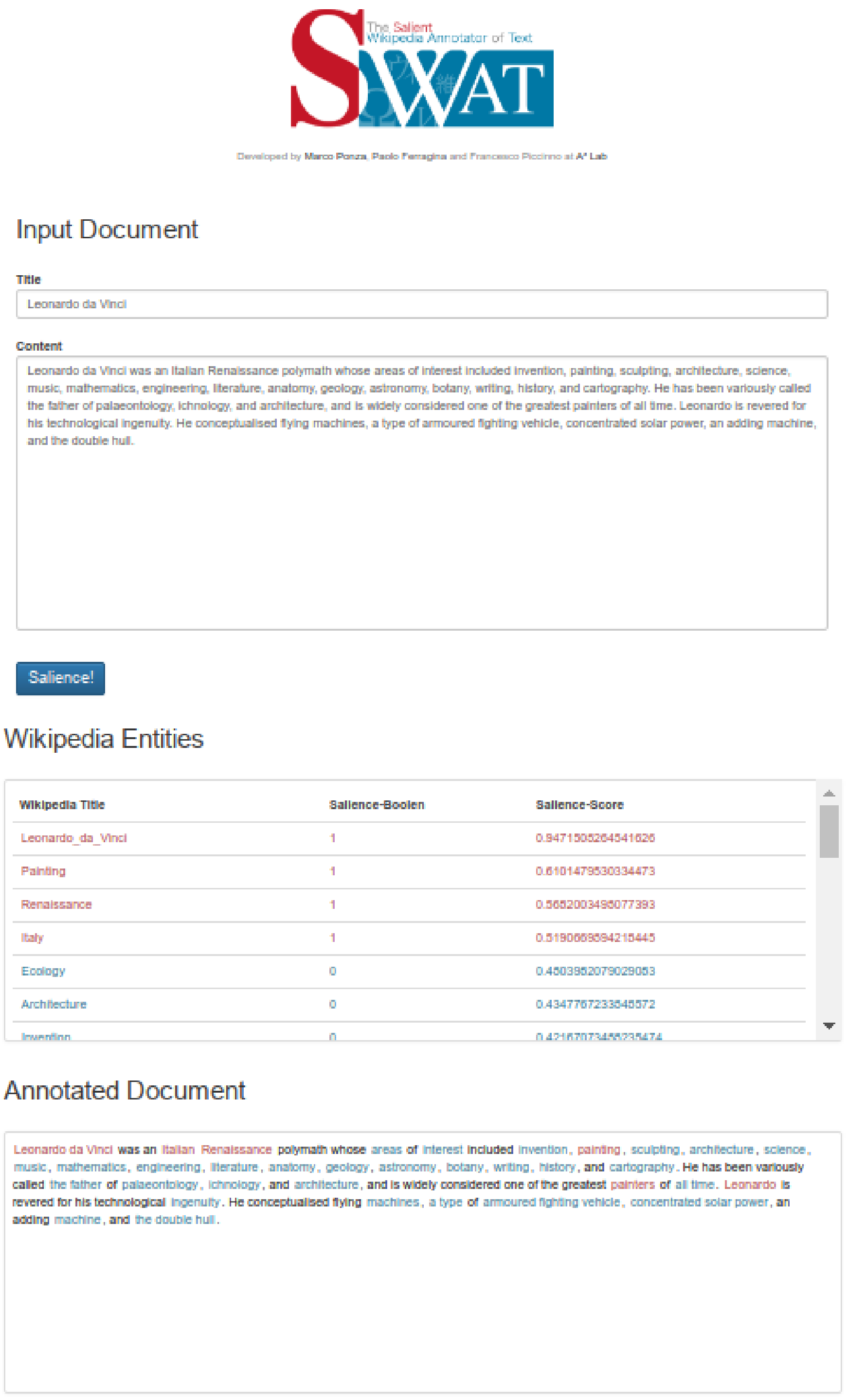
Name	Description	Type
title	Title of the document.	String
content	Content of the document.	String

Table A.2: Fields of the SWAT's JSON response.

Name	Description	Type
status	Status of the response.	String
annotations	List of extractions (see Table A.3).	List

Table A.3: Fields present in each object of `annotations` field in the JSON response.

Name	Description	Type
wiki_id	Wikipedia ID of the extracted entity.	Integer
wiki_title	Wikipedia title of the extracted entity.	String
salience_boolean	1 if the entity is salient, 0 otherwise.	Integer
salience_score	Score of relevance of the entity.	Float
spans	List of pairs of integers. Each pair contains the start (included) and end (excluded) offsets at character-level of the extracted entity in the input text.	List



Developed by Marco Ponza, Paolo Ferragina and Francesco Piccinno at A³ Lab

Input Document

Title

Leonardo da Vinci

Content

Leonardo da Vinci was an Italian Renaissance polymath whose areas of interest included invention, painting, sculpting, architecture, science, music, mathematics, engineering, literature, anatomy, geology, astronomy, botany, writing, history, and cartography. He has been variously called the father of palaeontology, ichnology, and architecture, and is widely considered one of the greatest painters of all time. Leonardo is revered for his technological ingenuity. He conceptualised flying machines, a type of armoured fighting vehicle, concentrated solar power, an adding machine, and the double hull.

Wikipedia Entities

Wikipedia Title	Salience-Boolen	Salience-Score
Leonardo_da_Vinci	1	0.9471505264541626
Painting	1	0.6101479530334473
Renaissance	1	0.5652003496077393
Italy	1	0.5190669594216445
Ecology	0	0.4503952079029053
Architecture	0	0.4347767233545572
Invention	0	0.42167073455235474

Annotated Document

Leonardo da Vinci was an Italian Renaissance polymath whose areas of interest included invention, painting, sculpting, architecture, science, music, mathematics, engineering, literature, anatomy, geology, astronomy, botany, writing, history, and cartography. He has been variously called the father of palaeontology, ichnology, and architecture, and is widely considered one of the greatest painters of all time. Leonardo is revered for his technological ingenuity. He conceptualised flying machines, a type of armoured fighting vehicle, concentrated solar power, an adding machine, and the double hull.

Figure A.1: The GUI of Swat prototype allows detecting and classifying Wikipedia entities from an input text. The box `Wikipedia Entities` shows the extracted entities with a boolean label, denoting salient (red) and non-salient (blue) entities, and ranked by their `Salience-Score`, namely XGboost's probability. The box `Annotated Document` shows the mentions extracted to their pertinent Wikipedia pages.

References

Abujabal, A., Saha Roy, R., Yahya, M., & Weikum, G. (2018). Never-ending learning for open-domain question answering over knowledge bases. In *Proceedings of WWW*.

Ackerman, M. S., Wulf, V., & Pipek, V. (2002). *Sharing Expertise: Beyond Knowledge Management*.

Agirre, E., Alfonseca, E., Hall, K. B., Kravalova, J., Pasca, M., & Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of NAACL-HLT*.

Anagnostopoulos, A., Broder, A. Z., Gabrilovich, E., Josifovski, V., & Riedel, L. (2011). Web page summarization for just-in-time contextual advertising. *Transactions on Intelligent Systems and Technology*.

Anthonisse, J. M. (1971). The rush in a directed graph. *Stichting Mathematisch Centrum. Mathematische Besliskunde*.

Aouicha, M. B., Taieb, M. A. H., & Ezzeddine, M. (2016). Derivation of "is a" taxonomy from wikipedia category graph. *Engineering Applications of Artificial Intelligence*.

Aprilius, W., Hansun, S., & Gunawan, D. (2017). Entity annotation wordpress plugin using tagme technology. *Telecommunication Computing Electronics and Control*.

Bairi, R. B., Carman, M., & Ramakrishnan, G. (2015). On the evolution of wikipedia: Dynamics of categories and articles. In *Proceedings of ICWSDM*.

Bakarov, A. (2018). A survey of word embeddings evaluation methods. *CoRR*.

Balasubramanian, N., Soderland, S., Etzioni, O., et al. (2012). Rel-grams: a probabilistic model of relations in text. In *Proceedings of AKBC-WEKEX*.

Balasubramanian, N., Soderland, S., Etzioni, O., et al. (2013). Generating coherent event schemas at scale. In *Proceedings of EMNLP*.

Balog, K., Azzopardi, L., & de Rijke, M. (2006). Formal models for expert finding in enterprise corpora. In *Proceedings of SIGIR*.

Balog, K., Azzopardi, L., & de Rijke, M. (2009). A language modeling framework for expert finding. *Information Processing and Management*.

Balog, K. & De Rijke, M. (2008). *Combining candidate and document models for expert search*. Technical report.

Balog, K., Fang, Y., de Rijke, M., Serdyukov, P., & Si, L. (2012). Expertise retrieval. *Foundation and Trends in Information Retrieval*.

Banerjee, S. & Mitra, P. (2016). Wikiwrite: Generating wikipedia articles automatically. In *Proceedings of IJCAI*.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., & Etzioni, O. (2007). Open information extraction from the web. In *Proceedings of IJCAI*.

Bansal, P., Bansal, R., & Varma, V. (2015). Towards deep semantic analysis of hashtags. In *Proceedings of ECIR*.

Basile, P., Caputo, A., & Semeraro, G. (2016). Entity linking for the semantic annotation of italian tweets.

Bast, H., Bäurle, F., Buchhold, B., & Haussmann, E. (2014). Semantic full-text search with Broccoli. In *Proceedings of SIGIR*.

Bast, H., Buchhold, B., & Haussmann, E. (2017). Overview of the triple scoring task at the WSDM cup 2017.

Baumel, T., Cohen, R., & Elhadad, M. (2014). Query-chain focused summarization. In *Proceedings of ACL*.

Bavelas, A. (1948). A mathematical model for group structures. *Human organization*.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*.

Berendsen, R., Balog, K., Bogers, T., van den Bosch, A., & de Rijke, M. (2013). On the assessment of expertise profiles. In *International Dutch-Belgian Workshop on IR*.

Bhagavatula, C. S., Noraset, T., & Downey, D. (2015). Tabel: entity linking in web tables. In *n Proceedings of ISWC*.

Bi, B., Ma, H., Hsu, B.-J. P., Chu, W., Wang, K., & Cho, J. (2015). Learning to recommend related entities to search users. In *Proceedings of WSDM*.

Bishop, M. C. (2016). *Pattern Recognition and Machine Learning*.

Blanco, R., Ottaviano, G., & Meij, E. (2015). Fast and space-efficient entity linking for queries. In *Proceedings of WSDM*.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching word vectors with subword information. *CoRR*.

Boldi, P. & Monti, C. (2016). Cleansing wikipedia categories using centrality. In *Proceedings of WWW*.

Boldi, P. & Vigna, S. (2004). The WebGraph framework I: Compression techniques. In *Proceedings of WWW*.

Boldi, P. & Vigna, S. (2014). Axioms for centrality. *Internet Mathematics*.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*.

Bollacker, K., Tufts, P., Pierce, T., & Cook, R. (2007). A platform for scalable, collaborative, structured information integration. In *Proceedings of IIWeb*.

Bordino, I., Morales, G. D. F., Weber, I., & Bonchi, F. (2013). From machu picchu to "rafting the urubamba river": anticipating information needs via the entity-query graph. In *Proceedings of WSDM*.

Breiman, L., Friedman, J., Olshen, R. A., & Stone, C. J. (1993). Classification and regression trees.

Brin, S. & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Proceedings of WWW*.

Brittain, J. M. (1975). *Information Needs and Application of the Results of User Studies*.

Bunescu, R. & Paşca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL*.

Cao, Y., Liu, J., Bao, S., & Li, H. (2005). Research on expert search at enterprise track of TREC 2005. In *Proceedings of TREC*.

Ceccarelli, D., Lucchese, C., Orlando, S., Perego, R., & Trani, S. (2013). Learning relatedness measures for entity linking. In *Proceedings of CIKM*.

Chabchoub, M., Gagnon, M., & Zouaq, A. (2016). Collective disambiguation and semantic annotation for entity linking and typing. In *Semantic Web Evaluation Challenge*.

Chakrabarti, S. (2002). *Mining the Web: Discovering knowledge from hypertext data*.

Chambers, N. & Jurafsky, D. (2009). Unsupervised learning of narrative schemas and their participants. In *Proceedings of ACL-AFNLP*.

Chambers, N. & Jurafsky, D. (2011). Template-based information extraction without the templates. In *Proceedings of ACL*.

Chen, D., Fisch, A., Weston, J., & Bordes, A. (2017). Reading wikipedia to answer open-domain questions. *CoRR*.

Chen, D. & Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*.

Chen, T. & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of SIGKDD*.

Cheng, X. & Roth, D. (2013). Relational inference for wikification. In *Proceedings of EMNLP*.

Chien, J.-T. & Chang, Y.-L. (2013). Hierarchical theme and topic model for summarization. In *Proceedings of MLSP*.

Chisholm, A. & Hachey, B. (2015). Entity disambiguation with web links. *Transactions of the Association of Computational Linguistics*.

Christensen, J., Soderland, S., Bansal, G., & Mausam (2014). Hierarchical summarization: Scaling up multi-document summarization. In *Proceedings of ACL*.

Christensen, J., Soderland, S., Etzioni, O., et al. (2013). Towards coherent multi-document summarization. In *Proceedings of NAACL-HLT*.

Cifariello, P., Ferragina, P., & Ponza, M. (2019). Wiser: A semantic approach for expert finding in academia based on entity linking. *Information Systems*.

Clark, K. & Manning, C. D. (2016). Improving coreference resolution by learning entity-level distributed representations. *CoRR*.

Cornolti, M., Ferragina, P., & Ciaramita, M. (2013). A framework for benchmarking entity-annotation systems. In *Proceedings of WWW*.

Cornolti, M., Ferragina, P., Ciaramita, M., Rüd, S., & Schütze, H. (2016). A piggyback system for joint entity mention detection and linking in web queries. In *Proceedings of WWW*.

Cucerzan, S. (2007). Large-scale named entity disambiguation based on wikipedia data. In *Proceedings EMNLP-CoNLL*.

Del Corro, L. & Gemulla, R. (2013). ClausIE: Clause-based open information extraction. In *Proceedings of WWW*.

Demartini, G. (2007). Finding experts using wikipedia. In *In Prceedings of FEW*.

Dietz, L., Xiong, C., & Meij, E. (2017). Overview of the first workshop on knowledge graphs and semantics for text retrieval and analysis (KG4IR). *Workshop on KG4IR*.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., & Zhang, W. (2014). Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings SIGKDD*.

Dunietz, J. & Gillick, D. (2014). A new entity salience task with millions of training examples. In *Proceedings of EACL*.

Durrett, G., Berg-Kirkpatrick, T., & Klein, D. (2016). Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of ACL*.

Erkan, G. & Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of SIGKDD*.

F. Fouss, A. P. & Saerens, M. (2005). A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation. In *Proceedings of WI*.

Fader, A., Soderland, S., & Etzioni, O. (2011). Identifying relations for open information extraction. In *Proceedings of EMNLP*.

Fang, Y., Si, L., & Mathur, A. P. (2010). Discriminative models of integrating document evidence and document-candidate associations for expert search. In *Proceedings of SIGIR*.

Ferragina, P., Piccinno, F., & Santoro, R. (2015). On analyzing hashtags in twitter. In *Proceedings of ICWSM*.

Ferragina, P. & Scaiella, U. (2012). Fast and accurate annotation of short texts with wikipedia pages. *IEEE Software*.

Fetahu, B., Markert, K., & Anand, A. (2015). Automated news suggestions for populating wikipedia entity pages. In *Proceedings of CIKM*.

Fogaras, D. & Rácz, B. (2005). Scaling link-based similarity search. In *Proceedings of WWW*.

Fox, E. A. & Shaw, J. A. (1994). Combination of multiple searches. *NIST special publication SP*.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*.

Gabrilovich, E. & Markovitch, S. (2007). Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of IJCAI*.

Galárraga, L., Heitz, G., Murphy, K., & Suchanek, F. M. (2014). Canonicalizing open knowledge bases. In *Proceedings of CIKM*.

Gambhir, M. & Gupta, V. (2017). Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*.

Gamon, M., Yano, T., Song, X., Apacible, J., & Pantel, P. (2013). Identifying salient entities in web pages. In *Proceedings of CIKM*.

Gandica, Y. C., Lambiotte, R., & Carletti, T. (2016). What can wikipedia tell us about the global or local character of burstiness? In *Proceedings of AAAI*.

Ganea, O.-E., Ganea, M., Lucchi, A., Eickhoff, C., & Hofmann, T. (2016). Probabilistic bag-of-hyperlinks model for entity linking. In *Proceedings of WWW*.

Gashteovski, K., Gemulla, R., & Del Corro, L. (2017). Minie: minimizing facts in open information extraction. In *Proceedings of EMNLP*.

Golub, G. H. & Reinsch, C. (1970). Singular value decomposition and least squares solutions. *Numerische Mathematik*.

Guo, Z. & Barbosa, D. (2014). Robust entity linking via random walks. In *Proceedings of CIKM*.

Harris, Z. S. (1954). Distributional structure. *Word*.

Hasan, K. S. & Ng, V. (2014). Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of ACL*.

Hasibi, F., Balog, K., & Bratsberg, S. E. (2016). Exploiting entity linking in queries for entity retrieval. In *Proceedings of ICTR*.

Hassan, S. & Mihalcea, R. (2011). Semantic relatedness using salient semantic analysis. In *Proceedings of AAAI*.

Haveliwala, T. H. (2002). Topic-sensitive pagerank. In *Proceedings of WWW*.

Heath, T., Motta, E., & Petre, M. (2006). Person to person trust factors in word of mouth recommendation. In *Proceedings of CHI*.

Hoffart, J., Seufert, S., Nguyen, D. B., Theobald, M., & Weikum, G. (2012). Kore: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of CIKM*.

Hoffart, J., Suchanek, F. M., Berberich, K., & Weikum, G. (2013). Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*.

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., & Weikum, G. (2011). Robust disambiguation of named entities in text. In *Proceedings of EMNLP*.

Hoffman, M. D., Blei, D. M., & Bach, F. R. (2010). Online learning for latent dirichlet allocation. In *Proceedings of NIPS*.

Hu, L., Wang, X., Zhang, M., Li, J., Li, X., Shao, C., Tang, J., & Liu, Y. (2015). Learning topic hierarchies for wikipedia categories. In *Proceedings of ACL*.

Hu, X., Zhang, X., Lu, C., Park, E. K., & Zhou, X. (2009). Exploiting wikipedia as external knowledge for document clustering. In *Proceedings of SIGKDD*.

Jeh, G. & Widom, J. (2002). Simrank: a measure of structural-context similarity. In *Proceedings of SIGKDD*.

Ji, H. & Grishman, R. (2011). Knowledge base population: Successful approaches and challenges. In *Proceedings of NAACL-HLT*.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. *CoRR*.

Jurafsky, D. (2000). *Speech & language processing*.

Kashyap, P., Mukherjee, S., Yates, A., & Weikum, G. (2018). Declare: Debunking fake news and false claims using evidence-aware deep learning. In *Proceedings of EMNLP*.

Katz, G., Ofek, N., Shapira, B., Rokach, L., & Shani, G. (2011). Using wikipedia to boost collaborative filtering techniques. In *Proceedings of RecSys*.

Kim, S. & Oh, A. (2016). Topical interest and degree of involvement of bilingual editors in wikipedia. In *Proceedings of AAAI*.

Kulkarni, S., Singh, A. V., Ramakrishnan, G., & Chakrabarti, S. (2009). Collective annotation of wikipedia entities in web text. In *Proceedings of SIGKDD*.

Lahiri, S., Mihalcea, R., & Lai, P.-H. (2017). Keyword extraction from emails. *Natural Language Engineering*.

Lample, G., Conneau, A., Denoyer, L., & Ranzato, M. (2017). Unsupervised machine translation using monolingual corpora only. *CoRR*.

Lamprecht, D., Dimitrov, D., Helic, D., & Strohmaier, M. (2016). Evaluating and improving navigability of wikipedia: A comparative study of eight language editions. In *Proceedings of OpenSym*.

Le, Q. & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of ICML*.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al. (2015). Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*.

Levy, O. & Goldberg, Y. (2014a). Dependency-based word embeddings. In *Proceedings of ACL*.

Levy, O. & Goldberg, Y. (2014b). Neural word embedding as implicit matrix factorization. In *Proceedings of NIPS*.

Li, H., Xu, J., et al. (2014). Semantic matching in search. *Foundations and Trends in Information Retrieval*.

Li, K., Zhang, J., Yao, C., & Shi, C. (2016). Automatic relation extraction from text: A survey. In *Proceedings of IIKI*.

Li, Y. & Yang, T. (2018). Word embedding for understanding natural language: A survey. In *Guide to Big Data Applications*.

Liben-Nowell, D. & Kleinberg, J. (2007). The link-prediction problem for social networks.

Lim, K. H. & Datta, A. (2013). Interest classification of twitter users using wikipedia. In *Proceedings of OpenSym*.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of research and development*.

Luo, G., Liu, Q., Sai, K. S. R., Bigongiari, D., Ke, Q., Nicolov, O. D., & Vadrevu, S. (2017). Automatic document summarization using search engine intelligence.

Luong, T., Socher, R., & Manning, C. (2013). Better word representations with recursive neural networks for morphology. In *Proceedings of CoNLL*.

Macdonald, C. & Ounis, I. (2006). Voting for candidates: Adapting data fusion techniques for an expert search task. In *Proceedings of CIKM*.

Macdonald, C. & Ounis, I. (2011). Learning models for ranking aggregates. In *Proceedings of ECIR*.

Maehara, T. et al. (2014). Computing personalized PageRank quickly by exploiting graph structures.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of ACL*.

Manning, C. D. (2011). Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *Proceedings of CICLing*.

Manning, D. C., Raghavan, P., & Schacetzel, H. (2008). Introduction to information retrieval.

Mausam, M. (2016). Open information extraction systems and downstream applications. In *Proceedings of IJCAI*.

McClosky, D., Surdeanu, M., & Manning, C. D. (2011). Event extraction as dependency parsing. In *Proceedings of ACL*.

McInnes, L. & Healy, J. (2017). Accelerated hierarchical density based clustering. In *Proceedings of ICDMW*.

Meij, E., Weerkamp, W., & De Rijke, M. (2012). Adding semantics to microblog posts. In *Proceedings of WSDM*.

Merchant, A., Shah, D., & Singh, N. (2016). In wikipedia we trust: A case study–extended abstract. In *Proceedings of AAAI*.

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (2013). *Machine learning: An artificial intelligence approach*.

Mihalcea, R. (2007). Using wikipedia for automatic word sense disambiguation. In *Proceedings of NAACL-HLT*.

Mihalcea, R. & Csomai, A. (2007). Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of CIKM*.

Mihalcea, R. & Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of EMNLP*.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*.

Milne, D. N. & Witten, I. H. (2008). An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceedings of AAAI*.

Minkov, E. & Cohen, W. W. (2012). Graph based similarity measures for synonym extraction from parsed text. In *Proceedings of TextGraphs*.

Mintz, M., Bills, S., Snow, R., & Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of IJCNLP*.

Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., et al. (2018). Never-ending learning. *Communications of the ACM*.

Mohler, M. & Mihalcea, R. (2009). Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of EACL*.

Moreira, C., Calado, P., & Martins, B. (2015). Learning to rank academic experts in the dblp dataset. *Expert Systems*.

Moro, A., Raganato, A., & Navigli, R. (2014). Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*.

Murphy, K. P., Weiss, Y., & Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of AUAI*.

Nakashole, N., Weikum, G., & Suchanek, F. (2012). Patty: a taxonomy of relational patterns with semantic types. In *Proceedings of EMNLP-CoNLL*.

Nanni, F., Zhao, Y., Ponzetto, S. P., & Dietz, L. (2016). Enhancing domain-specific entity linking in dh. *Proceedings of DH*.

Navigli, R. & Ponzetto, S. P. (2010). Babelnet: Building a very large multilingual semantic network. In *Proceedings of ACL*.

Navigli, R. & Ponzetto, S. P. (2012). Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*.

Nayeem, M. T. & Chali, Y. (2017). Extract with order for coherent multi-document summarization. *CoRR*.

Nguyen, D. B., Abujabal, A., Tran, N. K., Theobald, M., & Weikum, G. (2017a). Query-driven on-the-fly knowledge base construction. In *Proceedings of VLDB*.

Nguyen, D. B., Hoffart, J., Theobald, M., & Weikum, G. (2014). Aida-light: High-throughput named-entity disambiguation. *Proceedings of LDOW*.

Nguyen, D. B., Theobald, M., & Weikum, G. (2017b). J-reed: Joint relation extraction and entity disambiguation. In *Proceedings of CIKM*.

Ni, Y., Xu, Q. K., Cao, F., Mass, Y., Sheinwald, D., Zhu, H. J., & Cao, S. S. (2016). Semantic documents relatedness using concept graph representation. In *Proceedings of WSDM*.

OpenNLP, A. (2011). Apache software foundation. *URL http://opennlp. apache. org*.

Oramas, S., Espinosa-Anke, L., Sordo, M., Saggion, H., & Serra, X. (2016). Information extraction for knowledge base construction in the music domain. *Data & Knowledge Engineering*.

Ouyang, Y., Li, W., Zhang, R., Li, S., & Lu, Q. (2013). A progressive sentence selection strategy for document summarization. *Information Processing & Management*.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank citation ranking: Bringing order to the web.* Technical report.

Paranjpe, D. (2009). Learning document aboutness from implicit user feedback and document structure. In *Proceedings of CIKM*.

Pasca, M. (2016). The role of wikipedia in text analysis and retrieval. In *Proceedings of COLING*.

Pasca, M. (2018). Finding needles in an encyclopedic haystack: Detecting classes among wikipedia articles. In *Proceedings WWW*.

Pauls, A. & Klein, D. (2011). Faster and smaller n-gram language models. In *Proceedings of ACL*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*.

Pelleg, D., Moore, A. W., et al. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of ICML*.

Pellissier Tanon, T., Vrandečić, D., Schaffert, S., Steiner, T., & Pintscher, L. (2016). From freebase to wikidata: The great migration. In *Proceedings of WWW*.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Perozzi, B., Al-Rfou', R., & Skiena, S. (2014). Deepwalk: online learning of social representations. In *Proceedings of SIGKDD*.

Piccinno, F. & Ferragina, P. (2014). From tagme to wat: a new entity annotator. In *In International Workshop on ERD*.

Ponza, M., Del Corro, L., & Weikum, G. (2018a). Facts that matter. In *Proceedings of EMNLP*.

Ponza, M., Ferragina, P., & Chakrabarti, S. (2017a). A two-stage framework for computing entity relatedness in wikipedia. In *Proceedings of CIKM*.

Ponza, M., Ferragina, P., & Piccinno, F. (2017b). Document aboutness via sophisticated syntactic and semantic features. In *Proceedings of NLDB*.

Ponza, M., Ferragina, P., & Piccinno, F. (2018b). Swat: A system for detecting salient wikipedia entities in texts. In *CoRR*.

Ponzetto, S. P. & Strube, M. (2006). Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of NAACL-HLT*.

Quirk, R., Greenbaum, S., Leech, G., & Svartvik, J. (1985). *A Comprehensive Grammar of the English Language*.

Radinsky, K., Agichtein, E., Gabrilovich, E., & Markovitch, S. (2011). A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings WWW*.

Ratinov, L., Roth, D., Downey, D., & Anderson, M. (2011). Local and global algorithms for disambiguation to wikipedia. In *Proceedings of ACL*.

Raviv, H., Kurland, O., & Carmel, D. (2016). Document retrieval using entity-based language models. In *Proceedings of SIGIR*.

Richardson, M., Burges, C. J., & Renshaw, E. (2013). Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of EMNLP*.

Rizoiu, M.-A., Xie, L., Caetano, T., & Cebrian, M. (2016). Evolution of privacy loss in wikipedia. In *Proceedings of WSDM*.

Robertson, S., Zaragoza, H., et al. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*.

Romadhony, A., Widyantoro, D. H., & Purwarianti, A. (2016). Using relation similarity on open information extraction-based event template extraction. In *Proceedings of ICACSIS*.

Ross, B., Dado, M., Heisel, M., & Cabrera, B. (2018). Gender markers in wikipedia usernames. In *Wiki Workshop*.

Rothe, S. & Schütze, H. (2014). Cosimrank: A flexible & efficient graph-theoretic similarity measure. In *Proceedings of ACL*.

Rousseeuw, P. J. & Kaufman, L. (1990). *Finding groups in data*.

Ru, C., Tang, J., Li, S., Xie, S., & Wang, T. (2018). Using semantic similarity to reduce wrong labels in distant supervision for relation extraction. *Information Processing & Management*.

Sandhaus, E. (2008). The new york times annotated corpus. *Linguistic Data Consortium*.

Scaiella, U., Ferragina, P., Marino, A., & Ciaramita, M. (2012). Topical clustering of search results. In *Proceedings of WSDM*.

Schmitz, M., Bart, R., Soderland, S., Etzioni, O., et al. (2012). Open language learning for information extraction. In *Proceedings of EMNLP-CoNLL*.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys*.

See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *CoRR*.

Seyler, D., Dembelova, T., Del Corro, L., Hoffart, J., & Weikum, G. (2017). Knowner: Incremental multilingual knowledge in named entity recognition. *CoRR*.

Shen, W., Wang, J., & Han, J. (2015). Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*.

Shin, J., Wu, S., Wang, F., De Sa, C., Zhang, C., & Ré, C. (2015). Incremental knowledge base construction using deepdive. In *Proceedings of VLDB*.

Simov, K., Osenova, P., & Popov, A. (2017). Comparison of word embeddings from different knowledge graphs. In *Proceedings of LDK*.

Singer, P., Lemmerich, F., West, R., Zia, L., Wulczyn, E., Strohmaier, M., & Leskovec, J. (2017). Why we read wikipedia. In *Proceedings of WWW*.

Singh, P., Lin, T., Mueller, E. T., Lim, G., Perkins, T., & Zhu, W. L. (2002). Open mind common sense: Knowledge acquisition from the general public. In *Proceedings of OTM*.

Singhal, A. (2012). Introducing the knowledge graph: things, not strings.

Sorg, P. & Cimiano, P. (2011). Finding the right expert: Discriminative models for expert retrieval. In *Proceedings of KDIR*.

Sorokin, D. & Gurevych, I. (2017). Context-aware representations for knowledge base relation extraction. In *Proceedings of EMNLP*.

Sozio, M. & Gionis, A. (2010). The community-search problem and how to plan a successful cocktail party. In *Proceedings of SIGKDD*.

Speer, R. & Havasi, C. (2013). Conceptnet 5: A large semantic network for relational knowledge. In *The People's Web Meets NLP*.

Stanovsky, G., Dagan, I., & Mausam (2015). Open ie as an intermediate structure for semantic tasks. In *Proceedings of ACL*.

Strötgen, J. & Gertz, M. (2013). Multilingual and cross-domain temporal tagging. In *Proceedings of LREC*.

Strube, M. & Ponzetto, S. P. (2006). Wikirelate! computing semantic relatedness using wikipedia. In *Proceedings of AAAI*.

Suchanek, F. M., Kasneci, G., & Weikum, G. (2008). Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*.

Tan, C., Wei, F., Ren, P., Lv, W., & Zhou, M. (2017). Entity linking for queries by searching wikipedia sentences. *CoRR*.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of WWW*.

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., & Su, Z. (2008). Arnetminer: Extraction and mining of academic social networks. In *Proceedings of SIGKDD*.

Thater, S., Fürstenau, H., & Pinkal, M. (2010). Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of ACL*.

Trani, S., Lucchese, C., Perego, R., Losada, D. E., Ceccarelli, D., & Orlando, S. (2018). Sel: A unified algorithm for salient entity linking. *Computational Intelligence*.

Usbeck, R., Röder, M., Ngonga Ngomo, A.-C., Baron, C., Both, A., Brümmer, M., Ceccarelli, D., Cornolti, M., Cherix, D., Eickmann, B., et al. (2015). Gerbil: general entity annotator benchmarking framework. In *Proceedings of WWW*.

Van Gysel, C., de Rijke, M., & Kanoulas, E. (2016a). Learning latent vector spaces for product search. In *Proceedings of CIKM*.

Van Gysel, C., de Rijke, M., & Kanoulas, E. (2017). Semantic entity retrieval toolkit. In *In Workshop on Neu-IR*.

Van Gysel, C., de Rijke, M., & Worring, M. (2016b). Unsupervised, efficient and semantic expertise retrieval. In *Proceedings of the 25th International Conference on World Wide Web*.

Vatant, B. & Wick, M. (2012). Geonames ontology. *http://www.geonames.org/ontology/ontology_v3*.

Verhagen, M., Gaizauskas, R., Schilder, F., Hepple, M., Moszkowicz, J., & Pustejovsky, J. (2009). The tempeval challenge: identifying temporal relations in text. In *Proceedings of LREC*.

Vitale, D., Ferragina, P., & Scaiella, U. (2012). Classification of short texts by deploying topical annotations. In *Proceedings of ECIR*.

Voskarides, N., Meij, E., Reinanda, R., Khaitan, A., Osborne, M., Stefanoni, G., Kambadur, P., & de Rijke, M. (2018). Weakly-supervised contextualization of knowledge graph facts. In *Proceedings of SIGIR*.

Voskarides, N., Meij, E., Tsagkias, M., De Rijke, M., & Weerkamp, W. (2015). Learning to explain entity relationships in knowledge graphs. In *Proceedings of IJCNLP*.

Vrandečić, D. (2012). Wikidata: A new platform for collaborative data collection. In *Proceedings of WWW*.

Vrandečić, D. & Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of the ACM*.

Wagner, C., Garcia, D., Jadidi, M., & Strohmaier, M. (2015). It's a man's wikipedia? assessing gender inequality in an online encyclopedia. In *Proceedings of ICWSM*.

Wang, Y., Liu, S., Afzal, N., Rastegar-Mojarad, M., Wang, L., Shen, F., & Liu, H. (2018). A comparison of word embeddings for the biomedical natural language processing. *CoRR*.

Wasserman, S. & Faust, K. (1994). *Social network analysis: Methods and applications*.

Watts, D. J. & Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *Nature*.

Weikum, G., Hoffart, J., & Suchanek, F. M. (2016). Ten years of knowledge harvesting: Lessons and challenges. *IEEE Data Eng. Bull.*

West, R., Weber, I., & Castillo, C. (2012). A data-driven sketch of wikipedia editors. In *Proceedings of WWW*.

Wu, F. & Weld, D. S. (2010). Open information extraction using wikipedia. In *Proceedings of ACL*.

Xiong, S. & Luo, Y. (2014). A new approach for multi-document summarization based on latent semantic analysis. In *Proceedings of ISCID*.

Xu, R. & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*.

Yahya, M., Berberich, K., Elbassuoni, S., & Weikum, G. (2013). Robust question answering over the web of linked data. In *Proceedings of CIKM*.

Yan, Y., Okazaki, N., Matsuo, Y., Yang, Z., & Ishizuka, M. (2009). Unsupervised relation extraction by mining wikipedia texts using information from the web. In *Proceedings of NAACL-HLT*.

Yao, X. & Van Durme, B. (2014). Information extraction over structured data: Question answering with freebase. In *Proceedings of ACL*.

Yazdanian, R., Zia, L., & West, R. (2018). The elicitation of new users' interests on wikipedia. In *Wiki Workshop*.

Yeh, E. et al. (2009). WikiWalk: Random walks on Wikipedia for semantic relatedness. In *In International Workshop on GMNLP*.

Yimam, D. & Kobsa, A. (2000). DEMOIR: A hybrid architecture for expertise modeling and recommender systems. In *In Workshop on WETICE*.

Zhai, C. & Lafferty, J. (2017). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR*.

Zwicklbauer, S., Seifert, C., & Granitzer, M. (2016). Robust and collective entity disambiguation through semantic embeddings. In *Proceedings of SIGIR*.