INFORMATICA, 2023, Vol. 34, No. 3, 491–527
© 2023 *Vilnius University*
DOI: https://doi.org/10.15388/23-INFOR527

Automatically Assembling a Custom-Built Training Corpus for Improving the Learning of In-Domain Word/Document Embeddings

Yolanda BLANCO-FERNÁNDEZ[1,*], Alberto GIL-SOLLA[1],
José J. PAZOS-ARIAS[1], Diego QUISI-PERALTA[2]

[1] *atlanTTic Research Center for Telecommunication Technologies, University of Vigo, Spain*
[2] *Universidad Politécnica Salesiana de Cuenca, Ecuador*
e-mail: yolanda@det.uvigo.es, agil@det.uvigo.es, jose@det.uvigo.es, dquisi@ups.edu.ec

Received: March 2023; accepted: August 2023

Abstract. Embedding models turn words/documents into real-number vectors via co-occurrence data from unrelated texts. Crafting domain-specific embeddings from general corpora with limited domain vocabulary is challenging. Existing solutions retrain models on small domain datasets, overlooking potential of gathering rich in-domain texts. We exploit Named Entity Recognition and Doc2Vec for autonomous in-domain corpus creation. Our experiments compare models from general and in-domain corpora, highlighting that domain-specific training attains the best outcome.

Key words: embedding models, Named Entity Recognition, Doc2Vec, ad hoc corpus.

1. Introduction

The learning of word embeddings has gained momentum in many Natural Language Processing (NLP) applications, ranging from text document summarisation (Mohd *et al.*, 2020), fake news detection (Faustini and Covões, 2017; Silva *et al.*, 2020), and term similarity measure (Lastra *et al.*, 2019; Gali *et al.*, 2019) to sentiment classification (Rezaeinia *et al.*, 2019; Giatsoglou *et al.*, 2017; Park *et al.*, 2021), edutainment (Blanco *et al.*, 2020), Named Entity Recognition (Turian *et al.*, 2010; Gutiérrez-Batista *et al.*, 2018), classification tasks (Jung *et al.*, 2022) and personalization systems (Valcarce *et al.*, 2019), just to name a few. Most popular methods consider a large corpus of texts and represent each word with a real-valued dense vector, which captures its meaning assuming that words sharing common contexts in the input corpus are semantically related to each other (and consequently their respective word vectors are close in the vector space) (Mikolov *et al.*, 2013b). Drawing inspiration from such word representations, in last years document embeddings have emerged as a natural extension of word embeddings, by mapping variable-length documents (sentences, paragraphs or full documents) to vector representations. Their effectiveness has been remarkable in a wide diversity of tasks, such as text classification

[*]Corresponding author.

and sentiment analysis (Fu *et al.*, 2018; Le and Mikolov, 2014; Bansal and Srivastava, 2019), multi-document summarisation (Lamsiyah *et al.*, 2021; Rani and Lobiyal, 2022), forum question duplication (Lau and Baldwin, 2016), document similarity (Dai *et al.*, 2020), sentence pair similarity (Chen *et al.*, 2019), and even semantic relatedness and paraphrase detection (Logeswaran and Lee, 2018).

Mostly-adopted approaches to word and document embeddings leverage unsupervised learning methods from large collections of unlabelled documents that serve as training corpora in considering word-word co-occurrences. In the literature, commonly-used corpora compile a huge number of unrelated texts, such as a full collection of English Wikipedia, the Associated Press English news articles released from 2009 to 2015,[1] or a dataset of high quality English paragraphs containing over three billion words (Han *et al.*, 2013). Such collections lead to learning general-domain embedding models that do not perform well when working in a very specific domain (for example, on a particular historical event or a concrete medical discipline), whose common vocabulary is unlikely to be included in a generic corpus. As stated in Nooralahzadeh *et al.* (2018), "*domain-specific terms are challenging for general domain embeddings since there are few statistical clues in the underlying corpora for these items*". This idea was also previously stemmed from the results attained in Bollegala *et al.* (2015), Pilehvar and Collier (2016).

Bearing this limitation in mind, many researchers leveraged the findings of fields such as multi-task learning and transfer learning (Axelrod *et al.*, 2011) and adopted a mixed-domain training in two phases: First, a general domain corpus is used to train an embedding model, which is next trained incrementally with specialised documents that are related to the particular domain. Thanks to this continual training, the general knowledge of the first phase can be transfered to the second one in order to be exploited along with the lexical and semantic specificities in that domain (Liu *et al.*, 2015; Xu *et al.*, 2019). However, there also exist works that concluded that, in domains with abundant unlabelled texts, the domain-specific training is not improved with the transfer from general domains. As a running example in biomedicine, the authors of Gu *et al.* (2021) showed that "*domain-specific training from scratch substantially outperforms continual pretraining of generic language models, thus demonstrating that the prevailing assumption in support of mixed-domain pretraining is not always applicable*".

The benefits of resorting to only a domain-specific training from scratch have also been confirmed in other works (Nooralahzadeh *et al.*, 2018; Kim *et al.*, 2018; Lau and Baldwin, 2016). In particular, the authors of Nooralahzadeh *et al.* (2018) concluded that models learned from ad hoc corpora provide "*better results than general domain models for a domain-specific benchmark*", demonstrating besides that "*constructing domain-specific word embeddings is beneficial even with limited input data*". Actually, these approaches rely on small-sized ad hoc corpora, whose documents are gathered by hand and indiscriminately from publicly-available sources in the Internet (Chiu *et al.*, 2016; Nooralahzadeh *et al.*, 2018; Gu *et al.*, 2021; Cano and Morisio, 2017). Specifically, to the best of the authors' knowledge, existing approaches do not consider the relevance of a document (in the

[1] https://github.com/jhlau/doc2vec

particular domain) when deciding whether or not that text should be included in the ad hoc corpus. This process is obviously costly and clearly unfeasible without automatic assistance, in view of the myriad of possible domains/topics (and the huge amount of available documents on each of them). However, according to the results achieved in Chiu *et al.* (2016), the relevance of the specialised texts chosen as training documents is a critical parameter in learning of embedding models. Specifically, these researchers handled two ad hoc corpora including each one a different number of in-domain documents about biomedicine. Their results confirmed that the highest quality embedding models were learned from the smallest ad hoc corpus, proving that "*bigger corpora do not necessarily produce better biomedical domain word embeddings*". In other words, disregarding the suitability of the considered documents in the particular domain may distort the training.

Taking into account the above conditions, the interest and main contributions of the proposed approach can be summarised as follows:

- Ad hoc corpora enable learning successful embedding models in very specific domains (e.g. medicine, history or chemical engineering, to name a few), where the huge public generic datasets that are usually adopted fail to accurately model the peculiarities (and the particular vocabulary) of these specialised domains.
- The approach described in the paper automatically builds such corpora retrieving large amounts of candidate training texts from Internet sources and incorporating only those that are relevant in the context under consideration. For that purpose, NER facilities (to identify named entities in the input text) and Doc2Vec models (to assess the relationship between the input text and each of the retrieved candidate documents) are exploited.
- The only input required in the approach is a fragment of text representative of the context, making human assistance during the creation of the ad hoc corpus unnecessary and allowing to deal with any topic or domain, however specific they may be.
- The automatic procedure for building the resulting corpus greatly simplifies the hard work associated with traditional manual collection procedures, while providing more and better domain-specific documents.

This paper is organized as follows: Section 2 explores relevant works within the context of our approach to automatic custom training corpus creation. Section 3 focuses on the procedure devised to gather a set of in-domain candidate texts from the Internet, using a particular history event (the Battle of Thermopylae between Greeks and Persians in 480 BC) to illustrate the approach. The mechanism to validate the selection of *relevant* candidates to be incorporated into the automatically-built tailor-made corpus is detailed in Section 4. In this section, we also describe the tests conducted to evaluate the consistency of several Doc2Vec models, including a model trained on a general-domain corpus sourced from news articles of the Associated Press (AP), as well as in-domain models learned from ad hoc corpora. Finally, Section 5 concludes the paper and highlights further research directions.

2. Related Work

Our literature review is organized into three main sections. In Section 2.1, the focus is on prominent embedding models that lay the foundation for our research, whose learning in specialised scopes requires domain-specific training documents. Since our research contributes to the automatic generation of such kind of ad hoc corpora, Section 2.2 describes relevant related approaches for constructing custom datasets, emphasizing the key distinctions from our procedure. Our algorithm for selecting in-domain training documents starts by identifying named entities in the input text that contextualizes the specific theme for building the ad hoc dataset. Since named entities can have multiple possible interpretations, accurately distinguishing the associated meaning for each entity is crucial in this process. To achieve this, the commonly-adopted approaches to named entity disambiguation will be thoroughly reviewed in Section 2.3.

2.1. *Embedding Models*

The germ of learning language representations using models pre-trained on large collections of unlabelled texts springs from word embeddings such as Word2Vec (Mikolov *et al.*, 2013b) and GloVe (Pennington *et al.*, 2014). Word2Vec trains a model on the context of each word such that similar words have similar vector representations. Considering word-word co-occurrences, these embeddings capture semantics and meaning-related relationships (enabling, for instance, to detect that two words are similar or opposite, or that the pair of words *Spain* and *Madrid* have an analogous relation to *Canada* and *Ottawa*), along with syntax and grammar-related relationships (*have* and *had* are at same level as *are* and *were*). Word2Vec is a feed-forward neural network that learns vectors to improve its predictive ability, offering two different models: CBOW (where the goal is to predict a word based on the words in its context) and Skip-Gram (where the aim is to predict surrounding words given an input word) (Khatua *et al.*, 2019). This model suffers from two main weaknesses which, briefly, are mainly related to the impossibility of (i) dealing with words that do not appear in the training corpus, and (ii) considering different meanings for the same word.

- Regarding the first limitation, these kind of approaches are not able to embed Out-Of-Vocabulary (OOV) words unseen in the training corpus, which makes it impossible to deal with rare/unusual terms and misspelling. The embedding model FastText circumvents the OOV problem by working at character-n-gram level (Armand *et al.*, 2017).
- On the other hand, more sophisticated models have emerged in the literature which capture the contextualized meaning of words. In particular, models like ELMo (Petters *et al.*, 2018), GPT (Radford *et al.*, 2019) and BERT (Devlin *et al.*, 2019) enable to learn contextual relationships among words. For instance, in the sentences "*I have hit my head*" and "*The Head of school sent a message to the students*" the meaning of the word *head* depends on its left context in the first sentence (*I have hit my*) and on the right context in the second one (*of school sent a message to the students*). Bearing this motivation example in mind, some approaches moved from fixed word embeddings to contextualized models that consider both the sense of the next and previous words.

The main differences between the most popular contextualized embeddings (BERT, ELMo and GPT) are related to architectural internals: While ELMo relies on a Long Short-Term Memory (LSTM) model, GPT, BERT and its variants resort to a Transformer-based architecture (Reimers and Gurevych, 2019; Wang and Jay-Kuo, 2020). Details of both architectures, out of the scope of this paper, can be found in (Ethayarajh, 2019). Before the irruption of the latest BERT-based document embedding models, the commonly adopted approach was Paragraph Vector (also called Doc2Vec), the natural extension of Word2Vec for learning vector representations for pieces of variable-length texts (sentences, paragraphs and documents) (Le and Mikolov, 2014; Lau and Baldwin, 2016; Kim *et al.*, 2018; Bhattacharya *et al.*, 2022). Similar to Word2Vec, Doc2Vec works with two different models: Distributed Memory version of Paragraph Vector (PV-DM) and Distributed Bag Of Words version of Paragraph Vector (PV-DBOW). In both models the training enables to learn a vector for the initial text (called paragraph vector in Le and Mikolov, 2014), considering or not the Word2Vec embeddings of the single words depending on the approach, replicating such procedure in the prediction phase to provide a vector representing the paragraph/document.

The results described in Kim *et al.* (2018) highlighted that Doc2Vec outperformed previous sentence embeddings methods, ranging from simple approaches that use a weighted average of all the words in the document (Grefenstette *et al.*, 2013; Mikolov *et al.*, 2013a) to more sophisticated models like Skip-Thought (Kiros *et al.*, 2015) and Quick-Thoughts (Kiros *et al.*, 2018) that have attained good performance on diverse NLP tasks, such as semantic relatedness, paraphrase detection, image-sentence ranking and question-type classification (Kiros *et al.*, 2018). The main features of the above models are summarised in Table 1.

As noted in Section 1, many existing embedding models enable to fine-tune the training process on a new in-domain dataset for a concrete NLP task. However, as far as the authors of this paper know, there are no relevant approaches in the literature that automatically assemble such a dataset as a custom-built training corpus containing a large amount of relevant documents to capture meaningful domain-specific information. This would lead to more accurate embedding representations, which could improve the performance of existing (word-level and document- level) models. In this regard, it should be noted that

Table 1
Embedding models defined in the literature and their main features.

Embedding model	Word-level model	Sentence/document level model	Contextualized embeddings	Architecture
Word2Vec	✓	×	×	Feed-forward neural network
GloVe	✓	×	×	Count-based model
FastText	✓	×	×	Feed-forward neural network
ELMo	✓	×	✓	LSTM
GPT	✓	×	✓	Transformer
BERT	✓	×	✓	Transformer
Doc2Vec	✓	✓	✓	Feed-forward neural network
Skip-Thought	✓	✓	✓	Encoder-decoder
Quick-Thoughts	✓	✓	✓	Encoder-decoder

this paper is not about using an ad hoc corpus to train existing models with an ad hoc corpus in order to compare their respective performances and assess their strengths. Instead, the goal of this research is to devise a semantics-driven mechanism to automatically collect an in-domain dataset and verify that can lead to models with better performance that the ones trained with a generic corpus. To do so, the presented research considers a particular model (in this case, Doc2Vec) and a very specific application domain (a historical event like the Battle of the Thermopylae, as will be described in the validation scenario presented in Section 4).

Having tested this hypothesis with a Doc2Vec model, the viability and utility of the proposed semantics-driven mechanism are confirmed, and the door is open for its application in different scenarios involving other more sophisticated Transformer-based embeddings defined in the literature. This further goal is beyond the scope of the experimental validation presented in this paper, where the focus is on assessing the quality of a tailor-made training corpus rather than on quantifying the performance of the multiple models that could be learned from it.

2.2. *Automatic Creation of Corpora*

The web has long been considered a mega corpus with the potential to uncover new information across diverse fields (Crystal, 2011; Gatto, 2014). Consequently, numerous works in the literature focus on processing web-based corpora to find, extract, or transform information. This trend has intensified with the explosive growth of NLP research over the past decade, leading to extensive efforts in gathering both labelled and raw text corpora for discovering linguistic regularities, generating embeddings, and performing downstream tasks like classification, sentiment analysis, summarisation, or Q&A. Despite this importance, relevant results regarding automating the generation of such corpora remain scarce; there is hardly any research work related to automatic corpus creation.

The corpus construction approaches from the web that can be found in the literature primarily aim to support research and professional training in linguistic and translation fields. Typically, their objective is to create corpora for gaining an overview of a given language. Thus, these approaches mainly involve exploring the web to study pages based on their language rather than their content, resulting in general corpora rather than specialised ones. The most prevalent approach in these initiatives is the BootCat tool (Baroni and Bernardini, 2004), followed by successive refinements known as the WebBootCat web application (Baroni *et al.*, 2006) later marketed as SketchEngine (Kilgarriff *et al.*, 2014), relies on issuing a general set of search-engine queries involving domain-specific keywords to obtain focused collections of documents. This approach requires users to provide initial seeds (keywords) to begin the search. Subsequently, a large number of queries combining these seeds are issued against search engines like Google, Yahoo, or Bing, and the more relevant results are recovered to form the corpus.

However, these approaches primarily rely on web crawling followed by some linguistic processing to filter inadequate results. They work with literal keywords, querying for documents (web pages) containing those keywords, and repeat the process with the links contained in each page. They do not explore categories to classify the pages nor similarities among documents to establish their relevance for inclusion in the corpus or to trigger new searching processes or URL selection beyond those explicitly contained in the recovered documents. The text gathering procedure is somewhat coarse, and manual refinement is often necessary to guide these tools in the search for appropriate texts for the corpus to achieve a quality output. As a result, these projects lead to relatively small corpora, suitable for studying a language in a teaching environment but insufficient for training neural networks.

Similar approaches are used in projects creating corpora for linguistic and translation purposes, such as specialised health corpora (Symseridou, 2018) for instructing translation professionals in required abilities like locating terms, studying collocations, grammar, and syntax. The same approach is followed in Castagnoli (2015) to build corpora for health, law, and cell phone scenarios, and in Lynn *et al.* (2015), aimed at constructing linguistic corpora for less commonly used languages (e.g. Irish). In all these cases, the core algorithm is based on web crawling, with language being the main driving constraint for selecting pages to add or continue searching.

The evolution of these approaches has been significantly influenced by the explosive growth of the web and subsequent restrictions imposed by search engines regarding massive querying of the web (Barbaresi, 2013a). This led to the exploration of alternative ways to discover documents for the corpus, such as exploring social networks and blogging platforms (Barbaresi, 2013b), the Open Directory Project and Wikipedia (Barbaresi, 2014), or the Common Crawl platform, a free Internet crawling initiative (Smith *et al.*, 2013).

There are also automatic corpus-construction experiences centred not on the whole web but on closed repositories, aimed at filtering relevant documents for specific queries. These projects typically involve structured and well-known repositories, where the goal is to create information corpora restricted to characteristics specified by users, resembling more of a database query than web exploration. Their objectives focus on information analysis to guarantee compliance with restrictions rather than finding more similar texts from the current one. An example is Primpeli *et al.* (2019), which focuses on recovering text resources about e-commerce items from the WDC Product Data Corpus, extracted from the Common Crawl data repository. The compiled data, originally attached to product pages by e-commerce companies, forms an automatically created corpus used to group similar products in clusters. The quality of this corpus is confirmed by Peeters *et al.* (2020), Peeters and Bizer (2021), where embedding models were trained using such corpora. Zhang and Song (2022) utilizes information extracted from the same sources to feed various processes in the field of NLP, such as embedding generation or model training. Both scenarios share some similarities with ours as a corpus is created for training Machine Learning models. However, the documents collected in their research are located in a specific source, originating from an already available corpus containing semantic annotations, with no relevance to the subject being measured for such documents, and no new text is discovered from the analysed ones.

Numerous other research works claim automatic corpus creation in Machine Learning settings. For example, Zhang *et al.* (2021) creates a new corpus from the Amazon product dataset to train a new BERT model; Abacha and Dina (2016) automatically creates a corpus of equivalent pairs of questions (Textual Entailments) from the National Library of Medicine's database of questions (USA); Zanzotto and Pennacchiotti (2010) extracts pairs of entailments from Wikipedia by studying the successive historic revisions of some articles; and Zhou *et al.* (2022) builds a new corpus of Paraphrase Detection by refining existing corpora like the Stanford Natural Language Inference corpus (SNLI) and the Multi-Genre Natural Language Inference corpus (MNLI). However, to our knowledge, all these approaches focus on processing closed repositories to obtain a new corpus, with no exploration of the web (or large repositories like Wikipedia) to search for new unknown documents. Moreover, no examples exist of studying the relevance of documents for a given subject openly specified by the user; the working theme is already fixed at the creation of the project. In the first group, new documents are discovered simply by crawling (with the language constraint), while no new previously unknown document is discovered in the second group.

In summary, the literature includes several works on the automatic creation of corpora. On the one hand, there is an older research line centred on linguistic and translation fields, where approaches are relatively simple, exploring the web from user-provided seeds, and generally involving simple web crawlers primarily driven by the language of pages. On the other hand, more elaborate approaches are found in specific fields like health or e-commerce, as well as a significant number of cases also centered on Machine Learning model training. However, to our knowledge, all these approaches are focused on processing closed repositories to create a new corpus, with no exploration of the web (or large repositories like Wikipedia) to search for new unknown documents. In neither approach do examples exist of studying the relevance of documents for a given subject openly specified by the user, and no new documents are discovered beyond the initial corpus.

2.3. *Named Entity Disambiguation*

In the literature, named entity disambiguation (NED) is commonly defined as the process of determining the precise meaning or sense of a named entity within a given context. These named entities can be identified by well-known named entity recognition tools like DBpedia-Spotlight (Mendes *et al.*, 2011). More specifically, the goal of NED is to resolve ambiguity by associating the named entity with a specific concept within a semantic knowledge base. Previous studies have addressed the challenge of entity disambiguation through the utilization of statistical methods and rule-based approaches. These works take into account the contextual words surrounding the target named entity during the disambiguation process. However, they often neglect the semantic nuances of words and lack generalizability since the rules are typically specific to certain domains (An *et al.*, 2020; Songa *et al.*, 2019).

Subsequently, more advanced mechanisms emerged, such as the methods based on entity features. In these approaches, when an entity possesses multiple interpretations, inconsistent entities are filtered out by assessing their semantic similarity. The disambiguation process considers the semantic attributes of the entity, the contextual information surrounding the entity, and even its frequency of occurrence in the processed text. Notably, these methods leverage contextual embedding models, which assign different vector representations to entities with the same spelling based on their specific meanings within each context. To achieve this, entity features-based disambiguation methods typically obtain the contextual embedding vector of the target entity. They then calculate the semantic distance between this vector and the embedding vectors of each candidate entity to effectively disambiguate and remove any ambiguous entities (Barrena *et al.*, 2015; Zwicklbauer *et al.*, 2016). However, despite the efficacy of this approach, it disregards the structural characteristics of the knowledge base in which the target entity is situated, such as the interconnections between entities. Consequently, it fails to capture the global semantic features of each entity (Adjali *et al.*, 2020). Additionally, this disambiguation method requires large training corpus to learn an embedding model.

To address this challenge, recent studies have turned to deep neural networks for entity disambiguation. Specifically, neural network-based approaches have gained popularity by incorporating the subgraph structure features of knowledge bases. These features are utilized as inputs to graph neural networks, enabling the disambiguation of entities within the knowledge base (Ma *et al.*, 2021). Various methods have been explored, including convolutional and recurrent neural networks, as well as LSTM networks, to disambiguate entities based on extracted associations among them (Geng *et al.*, 2021; Phan *et al.*, 2017).

Transformer-based language models have also demonstrated significant promising in capturing complex linguistic knowledge, leading researchers to employ attention mechanisms to obtain contextual embedding vectors for each entity and consider coherence between entities for joint disambiguation (Ganea and Hofmann, 2017). Furthermore, graph neural networks have been trained to acquire entity graph embeddings that encode global semantic features, subsequently transferred to statistical models to address entity ambiguity. While these approaches demonstrate potential in achieving human-level performance in entity disambiguation, they often require substantial amounts of training data and computational resources (Hu *et al.*, 2020). Therefore, challenges persist in optimizing these models and reducing their reliance on in-domain training datasets, which may not always be readily available, especially in highly specific or specialised domains like those handled in our ad hoc corpus generation approach. In simple terms, these models are not suitable for our purposes because they require training with ad hoc corpora, which is precisely what our work seeks to achieve, that is, automatically gathering collections of in-domain texts that were previously absent in the literature.

While we acknowledge the positive outcomes achieved by existing approaches in entity disambiguation within recent literature, their complexity and requirements, such as domain-specific training datasets and high computational demands, surpass the needs of our ad hoc corpus generation algorithm. In contrast, we employ a simpler yet effective mechanism, as evidenced by the obtained results, to identify the right entities in the given initial text. Details will be given in Section 3.2.

3. How to Build a Domain-Specific Training Corpus

The candidate documents to be incorporated into the automatically-built ad hoc corpus (for training the embedding models) are gathered from the Internet by a procedure that has been implemented in Python and made freely available in a GitHub repository (https://github.com/gssi-uvigo/Plethora). Specifically, the approach takes as input an initial text and retrieves Wikipedia articles that have a meaningful relationship with it through an algorithm that can be outlined as follows:

- First, the NER facilities provided by the DBpedia Spotlight tool are exploited to identify DBpedia named entities present in the input text (denoted as DB-SL entities). In addition, the approach searches for other semantically related entities that share some common features with these DB-SL entities (e.g. semantic topics and categories or wikicats). This step is addressed in Sections 3.1, 3.2 and 3.3.
- Next, the goal is to retrieve (and preprocess to remove irrelevant information) Wikipedia articles in which the previously identified entities are mentioned, as described in Sections 3.4 and 3.5.
- Finally, the retrieved texts that are actually relevant (according to the relatedness measured between each of them and the input text) are incorporated into the ad hoc corpus. As detailed in Section 3.6, this stage of the algorithm is driven by a semantic similarity metric based on a Doc2Vec embedding model.

Before delving into each phase of our algorithm, it is essential to justify the usage of Wikipedia in our research. Specifically, we prioritize retrieving texts from this source due to several compelling advantages: (i) Wikipedia serves as an extensive repository encompassing information about any subject, and it includes entries for relevant individuals, places, or events; (ii) DBpedia provides a wealth of semantic information about these entries, enhancing the depth and context of our analysis; and (iii) there is a well-established and reliable mechanism to follow links between these repositories, and even connect them to others, which facilitates the discovery and retrieval of new documents.

While our approach to constructing ad hoc corpora is equally effective for texts from Wikipedia or any other source, there are additional remarkable features of this information repository that make it particularly suitable: documents cover a wide range of topics and disciplines; these articles are written and reviewed by a committed community of volunteer contributors; and it is constantly updated, reflecting recent advances in different fields of knowledge. Further evidence supporting the quality, representativeness, and significance of the texts within this online encyclopedia is demonstrated by the use of large corpora of articles extracted from Wikipedia in Transformer architectures. These architectures have garnered remarkable achievements in the field of NLP by employing such corpora for pre-training their base models, enabling them to acquire extensive language knowledge and a broad contextual understanding. This initial pre-training phase primes the models before they are fine-tuned for specific NLP tasks, underscoring the significance and value of the texts extracted from Wikipedia in fostering the advancement of sophisticated language models.

3.1. *Strategy and Sources of Information*

As the aim is to build an ad hoc corpus, it is necessary to define some way of characterising the topic on which this tailor-made dataset should be based (i.e. a seed describing the context of interest). For that purpose, a short initial text is used, representative of the thematic to which all the document in the corpus should be more or less related. In other words, the goal is to search the Internet for documents with some kind of relationship to this initial text.

All along this document, the following initial text will be used.

> The Battle of Thermopylae was fought between an alliance of Greek city-states, led by King Leonidas of Sparta, and the Persian Empire of Xerxes I over the course of three days, during the second Persian invasion of Greece. It took place simultaneously with the naval battle at Artemisium, in August or September 480 BC, at the narrow coastal pass of Thermopylae ("The Hot Gates"). The Persian invasion was a delayed response to the defeat of the first Persian invasion of Greece, which had been ended by the Athenian victory at the Battle of Marathon in 490 BC over the Persian forces led by Darius I. By 480 BC Xerxes had amassed a huge army and navy, and set out to conquer all of Greece. The Athenian politician and general Themistocles had proposed that the allied Greeks block the advance of the Persian army at the pass of Thermopylae, and simultaneously block the Persian navy at the Straits of Artemisium. A Greek force of approximately 7,000 men marched north to block the pass in the middle of 480 BC. The Persian army, alleged by the ancient sources to have numbered over one million, but today considered to have been much smaller (various figures are given by scholars, ranging between about 100,000 and 150,000), arrived at the pass in late August or early September. The vastly outnumbered Greeks held off the Persians for seven days (including three of battle) before the rear-guard was annihilated in one of History's most famous last stands. During two full days of battle, the small force led by Leonidas blocked the only road by which the massive Persian army could pass. After the second day, a local resident named Ephialtes betrayed the Greeks by revealing that a small path led behind the Greek lines. Leonidas, aware that his force was being outflanked, dismissed the bulk of the Greek army and remained to guard their retreat with 300 Spartans, 700 Thespians, and 400 Thebans, fighting to the death.

This 1926 character-long text (hereafter denoted as T_0) is related to the Battle of Thermopylae among Greeks and Persians in 480 BC (Blanco *et al.*, 2020). That is, the aim is to compose a corpus related to the Greco-Persian wars, starting from a brief text related to the second Persian invasion of Greece and specifically to the famous and inspiring Battle of Thermopylae.

The approach conducted in this paper to discover documents leverages the Semantic Web and the Linked Open Data (LOD) (Oliveira *et al.*, 2017) initiatives, which form a global repository of interrelated knowledge with a multitude of structured data to study their relationships and obtain new information from the available one. Thus, the core of the procedure is based on identifying relevant entities in the initial text (e.g. people, locations, events…), discovering categories in which these entities are classified and, finally, gathering other entities also classified in those categories. For each entity discovered, its description can be retrieved from the LOD repositories, becoming a new candidate text to be included in the domain-specific corpus.

To delimit this work, the initial source of the data considered is Wikipedia and its structured counterpart, the DBpedia (Lehmann *et al.*, 2012). Given the vast amount of information available in these repositories, we leverage the existing capability to freely query them through well-known endpoints by SPARQL (SPARQL Protocol And RDF Query Language) queries. In particular, SPARQL is a language explicitly designed for retrieving data stored in RDF format through queries to repositories like DBpedia. DBpedia is not an isolated information repository but allows establishing links to other well-known datasets to enhance query results, such as YAGO (Pellissier *et al.*, 2020) and WikiData (Ismayilov *et al.*, 2015) that are extensively used in this work.[2] In sum, SPARQL plays a crucial role in the Semantic Web and Linked Open Data (LOD) initiatives due to its remarkable capabilities in pattern searching and result filtering based on specified conditions, enabling efficient access to information within semantic repositories.

Regarding the categories in which to classify the DB-SL entities identified in T_0, it is interesting to highlight two properties that are frequently used in metadata descriptions to link Wikipedia pages to categories (and their members).

- On the one hand, the `dct:subject` property links pages to categories in the Wikipedia categorization system. Each category is usually composed of several words joined by an underscore character (`Traitors_in_history`, `People_of_the_Greco-Persian_Wars`) and may represent classifications by page contents or by administrative goals (`Wikipedia_administration_templates`, `Articles_with_broken_or_outdated_citations`…).
- On the other one, through the `rdf:type` property, pages (and the entities they represent) are associated to YAGO wikicats, some classes of the YAGO ontology reflecting the Wikipedia categorization system. The format of such wikicats is `WikicatW1W2...Wn`, that is, the `Wikicat` string followed by a set of words concatenated where each word starts with an uppercase letter and continues with lowercase ones (e.g. `BattlesInvolvingAthens`, `LocationsInGreekMythology`).

As will be described in the next sections, both properties are exploited in the paper as they are significant sources of information on the subject of a document, thus helping to discover new data and to assess its relevance.

3.2. *Identifying DBpedia Entities in the Initial Text*

The identification of the relevant entities present in the initial text T_0 is based on the Named Entity Recognition capabilities provided by DBpedia Spotlight (DB-SL), which enable to move from raw text (strings referred as to surface forms) to structured data (the URLs of the corresponding DBpedia entities). Through several sophisticated procedures (such as entity detection, name resolution, candidate disambiguation…), DB-SL establishes the association among surface forms and DBpedia entities depending on the context: the same text can lead to different entities, and different surface forms can lead to the same entity.

To this aim, DB-SL provides both a web application interface available online and a well-known endpoint running an API to programmatically access the service remotely.[3] This last option is the most interesting one since the aim of this work is develop an automatic service that should be as autonomous as possible. However, this official service

[2]Of course, even though other sources of information could be easily explored through the appropriate study of their URL formats, it will be shown that the components and tools involved in this research are representative enough of the potentialities of this approach.

[3]http://model.dbpedia-Spotlight.org/en/annotate

rejects bulk queries as it is only provided for testing purposes, so to speed up the execution it is advisable to install a local copy of DBpedia-Spotlight using a Docker image, for instance, provided by its creators.[4]

So, the text T_0 is sent to the local DB-SL deployment to identify the relevant DBpedia entities contained in it. As a result, some DBpedia entities are retrieved, including the surface forms, links to DBpedia pages associated to them, and some additional information (e.g. the candidates for disambiguation and their rankings). The set of DBpedia entities obtained is denoted as $DE(T_0)$ in Eq. (1) (the mathematical notation adopted throughout the description of the approach can be found in Appendix A):

$$DE(T_0) = \{e_i/e_i \text{ is a DBpedia entity present in } T_0 \text{ according to DB-SL}\}. \tag{1}$$

In this example, **18 entities** were detected in the text0 T_0.[5]

Sometimes, DB-SL is not able to properly disambiguate candidates and provides some wrong entity associations in the results. In the example, for instance, the `First_French_Empire` entity is incorrectly identified by DB-SL from the word `empire` of the initial text. In other cases, some entities not denoting real persons, locations, events... but concepts are identified (e.g. `Battle`). These cases can lead to a huge amount of useless data that will be discarded later, but this usually introduces a heavy computation load and disk requirements that it would be convenient to avoid. In these circumstances, a named entity disambiguation method becomes necessary.

In spite of the notable performance achieved by the existing approaches in named entity disambiguation described in Section 2.3, their complexity and requirements, such as the necessity of domain-specific training datasets that are hard to find and high computational demands, go beyond the needs of our ad hoc corpus generation algorithm. Depending on such custom in-domain datasets for disambiguating named entities in a work like ours, which specifically aims to construct tailor-made ad hoc corpora that are missing in the literature, would be impractical. In these circumstances, we have opted to employ a simpler yet effective mechanism to identify the correct entities in the initial text T_0.

In particular, our mechanism leverages the semantic attributes, such as wikicats and subjects, associated with each entity in DBpedia and other linked repositories. Indeed, our approach specifically targets the identification of overlaps between the attributes of the target entity and those of other entities within T_0. The underlying assumption is that the correct interpretation of a target named entity will be categorized under the same wikicats and subjects as the other named entities identified in T_0. By considering these shared attributes, the mechanism increases the likelihood of accurately disambiguating the target entity within its given context.

Applying this procedure, some entities are discarded, such as `First_French_Empire` or `Battle`. Finally, after this filtering, only **10 entities** remained and were used

[4] Available at https://hub.docker.com/r/dbpedia/dbpedia-Spotlight

[5] In the following, some numbers will be provided, which are related to this default text. These numbers are continuously changing in a minor way, as Wikipedia pages are frequently added, removed, or modified, and new categories are constantly being created.

in the following phases (`Themistocles`, `Ephialtes_of_Trachis`, `Thespiae`, `Battle_of_Artemisium`, `Battle_of_Marathon`, `Leonidas_I`, `Darius_I`, `Sparta`, `Xerxes_I`, `Battle_of_Thermopylae`). As it can be seen, all of them have a strong relationship with the initial text. So, Eq. (1) is refined resulting in Eq. (2):

$$DE(T_0) = \{e_i / e_i \text{ is a DBpedia entity present in } T_0 \text{ according to DB-SL} \wedge \\ e_i \text{ is related to other } T_0 \text{ entities through wikicats/subjects}\}. \tag{2}$$

This approach has proven to be sufficient as any errors in the disambiguation process only have a limited impact on our algorithm. As explained throughout the paper, if we fail to identify any entity, we consider tangentially related documents to T_0 as alternatives, but these documents are subsequently discarded in the following steps of the algorithm. Essentially, misidentifying an entity may cause a delay in constructing the ad hoc corpus (which is not critical since real-time requirements are absent), but it will not result in irrelevant documents (according to the specific theme of T_0) being included within this custom dataset.

3.3. *Selecting All Relevant Wikicats that Characterise the Initial Text*

With the goal of finding new documents that are significantly related to the initial text, T_0 is characterised with the set of wikicats linked to all the entities discovered in the previous step. As shown in Eq. (3), the set of wikicats that characterise a given entity e is denoted as $WK(e)$.

$$WK(e) = \{wk_j / wk_j \text{ is a wikicat associated to the entity } e\}. \tag{3}$$

To carry out this characterisation process, it is necessary to analyse the property `rdf:type` of each entity e, as wikicats are the values of this property belonging to the `yago` namespace and starting with the string "`Wikicat`", as depicted in the next example:

$$\texttt{entity} \xrightarrow{\texttt{rdf:type}} \texttt{yago:Wikicat5th-centuryBCRulers}.$$

This `rdf:type` property is returned by DB-SL, but most of the times incompletely, so it is necessary to resort to the original information repository to collect extended structural descriptions of the identified entities, including the categories to which they belong to. In particular, the well-known properties `rdf:type` and `dct:subject` are used to discover the categories to which a given entity is associated. As an example, note the SPARQL query launched against the DBpedia well-known endpoint[6] to complete the information related to the entity `Leonidas_I`:

[6]https://dbpedia.org/sparql

```
SELECT group_concat(distinct ?type; separator=';') WHERE {
     VALUES ?uri {<http://dbpedia.org/resource/Leonidas_I>} .
     ?uri rdfs:label ?label; rdf:type ?type .
     FILTER(regex(?type,'http://dbpedia.org/class/yago/Wikicat')) .
     FILTER(lang(?label) = 'en')
}
```

Simple wikicats consisting of a single word are eliminated – e.g. `WikicatKings` or `WikicatBattle` – as they mostly lead to very general concepts, not sufficiently related to the initial text T_0. As mentioned before, these cases are likely to lead to a large number of URLs that would be ranked lower later and discarded, introducing unnecessarily huge computation requirements. Any relevant URLs reached from such wikicats are likely to be also obtained from other more significant wikicats. So, Eq. (3) is refined resulting in Eq. (4):

$$WK(e) = \{wk_j / wk_j \text{ is a “not simple” wikicat associated to the entity } e\}. \tag{4}$$

Next, the set $WK(T_0)$ is created as the aggregation of wikicats (removing duplicates) coming from the different entities contained in $DE(T_0)$. This is the set of relevant wikicats that characterise T_0, as shown in Eq. (5):

$$WK(T_0) = \bigcup_i WK(e_i), \quad \forall e_i \in DE(T_0). \tag{5}$$

In the example, **42 different wikicats** were detected associated to the **10 entities** identified in the input text.

Sometimes, depending on the entities found by DB-SL, a large number of wikicats are collected in this phase. In order not to disperse the search, at this time the user has the possibility to discard some of those wikicats (see Fig. 1) if they are not meaningful in the target context, keeping only the selected set of wikicats for the following steps.[7]

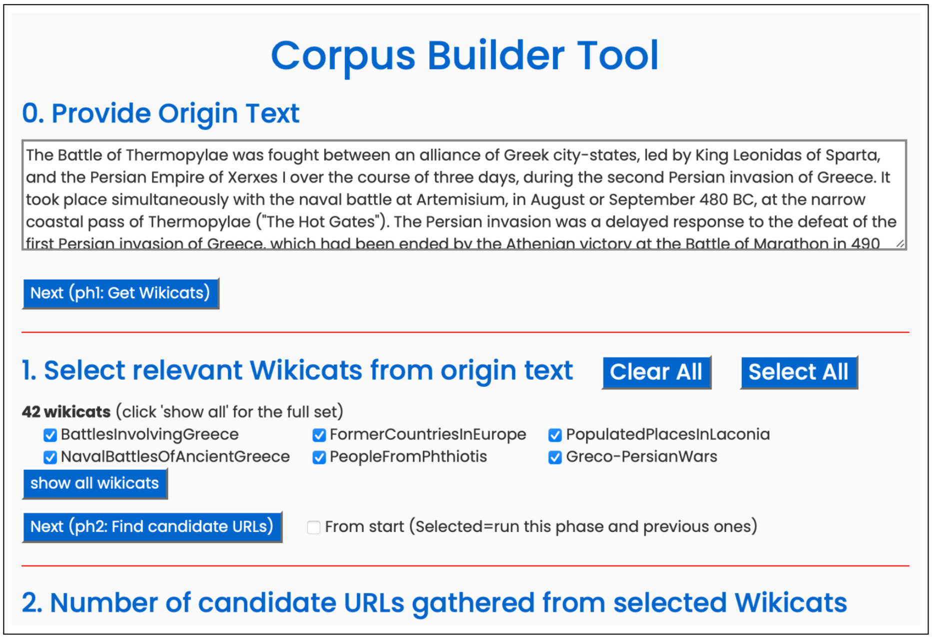
3.4. *Discovering New URLs Associated to the Relevant Wikicats*

So far, a number of entities have been identified in T_0 and some of their properties (wikicats and subjects) have been obtained. Now, it is time to exploit the rich capabilities of the LOD infrastructure to perform the reverse operation, that is, to collect objects (identified by URLs) that meet some requirements. For this purpose, well-known repositories will be explored to discover web pages characterised with the same tags that describe the initial text T_0.

First, for each wikicat in $WK(T_0)$, the DBpedia repository is queried to gather all the known DBpedia entities that are associated with it (denoted as $U_{DB}(w_k)$ in Eq. (6)). As each DBpedia entity is linked to a Wikipedia page, the set of Wikipedia URLs about entities tagged with that wikicat is also retrieved in this process. That is:

$$U_{DB}(w_k) = \{u_j / u_j \text{ is a URL tagged with wikicat } w_k \text{ according to DBpedia}\}. \tag{6}$$

[7] This is just an optimization to speed up the process.


Corpus Builder Tool

0. Provide Origin Text

The Battle of Thermopylae was fought between an alliance of Greek city-states, led by King Leonidas of Sparta, and the Persian Empire of Xerxes I over the course of three days, during the second Persian invasion of Greece. It took place simultaneously with the naval battle at Artemisium, in August or September 480 BC, at the narrow coastal pass of Thermopylae ("The Hot Gates"). The Persian invasion was a delayed response to the defeat of the first Persian invasion of Greece, which had been ended by the Athenian victory at the Battle of Marathon in 490

1. Select relevant Wikicats from origin text

42 wikicats (click 'show all' for the full set)

- [✓] BattlesInvolvingGreece
- [✓] FormerCountriesInEurope
- [✓] PopulatedPlacesInLaconia
- [✓] NavalBattlesOfAncientGreece
- [✓] PeopleFromPhthiotis
- [✓] Greco-PersianWars

☐ From start (Selected=run this phase and previous ones)

2. Number of candidate URLs gathered from selected Wikicats

Fig. 1. Snapshot of the corpus builder tool developed to explore and identify wikicats that are relevant in the context of the initial text T_0 (available at `https://github.com/gssi-uvigo/Plethora`).

To fetch this set of URLs, the following SPARQL query is sent to the well-known DBpedia endpoint[8] (being `Wikicat5th-centuryBCRulers` the wikicat searched in this example):

```
SELECT ?urlDBpedia ?urlWikipedia WHERE {
    ?urlDBpedia rdf:type yago:Wikicat5th-centuryBCRulers.
    ?urlDBpedia foaf:isPrimaryTopicOf ?urlWikipedia
}
```

The approach considers the `primaryTopic` property as this relationship is the one that leads directly to the Wikipedia page corresponding to the DBpedia entity (if any). If this property is not defined, the URL is discarded as it does not correspond to a Wikipedia page (since the interest does not lie in the URL of the DBpedia entity but in the text of its corresponding Wikipedia page). This text is the training document that will be added to the ad hoc corpus if is related enough to the initial text.

In addition, Wikidata (Vrandeĉić and Krötzsch, 2014; Yoo and Jeong, 2020) is also queried to gather all the Wikipedia pages related to the components of the wikicat name. Wikidata is the central repository of structured information for all the projects of the Wikimedia Foundation, storing more than 92 million data items (text, images, dates, ...) accessible by SPARQL queries. Same as Wikipedia, Wikidata is aimed at a crowdsourced data

[8]https://dbpedia.org/sparql

acquisition, being freely editable by people or programs, not only regarding contents, but also data structure. To fetch this second set of URLs, the following SPARQL query is made to the Wikidata well-known access endpoint[9] (being this time `Greco-PersianWars` the wikicat searched in the query):

```
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>
PREFIX mwapi: <https://www.mediawiki.org/ontology#API/>
     SELECT * WHERE {
       SERVICE wikibase:mwapi {
         bd:serviceParam wikibase:api 'Search' .
         bd:serviceParam wikibase:endpoint 'en.wikipedia.org' .
         bd:serviceParam mwapi:language "en" .
         bd:serviceParam mwapi:srsearch 'Greco-PersianWars' .
         ?title wikibase:apiOutput mwapi:title .
       }
}
```

This query provides a second set of URLs (denoted as $U_{WK}(w_k)$ in Eq. (7)), usually larger than the first one, although composed of less reliable documents.

$$U_{WK}(w_k) = \{u_j/u_j \text{ is related to the wikicat } w_k \text{ according to Wikidata}\}. \tag{7}$$

The second query permits to collect some interesting documents tightly related to the application scenario that, by any reason, have not been tagged by users with the set of characterising wikicats (may be even they are tagged with some similar wikicat that has not been retrieved in the first phase, e.g. `NavalBattlesInvolvingGreece`). In any case, less reliable documents will obtain a low rank in the following steps, and they will be discarded.

Finally, both sets of URLs (represented in Eqs. (6) and (7)) are joined (removing duplicates) resulting in Eq. (8):

$$U_{DW}(w_k) = U_{DB}(w_k) \cup U_{WK}(w_k). \tag{8}$$

At this point, $U(T_0)$ is defined as the set of URLs that are associated (in DBpedia or Wikidata) with some wikicat included in $WK(T_0)$ (that is, URLs of pages that may have a strong relationship with T_0), as shown in Eq. (9):

$$U(T_0) = \left\{u_j/\exists w_k \in WK(T_0) \text{ such that } u_j \in U_{DW}(w_k)\right\}. \tag{9}$$

In the example, **90735 different URLs** were identified from the **42 wikicats** collected. All of them corresponded to Wikipedia pages that were likely related to the context defined by the initial text T_0.

[9]https://query.wikidata.org/sparql

3.5. *Fetching and Cleaning Discovered URLs*

Every URL u_j included in $U(T_0)$ is now downloaded and cleaned (markup, styling, references, graphical items, etc., are removed) to become a candidate text to be included in the corpus.[10] Actually, only those documents with a minimum text length (currently 300 bytes) are considered as candidate texts, just because short texts usually denote meaningless pages, unlikely to contain DBpedia entities (for instance, disambiguation pages). In this regard, note that in the example considered in the paper, **83919 texts** were downloaded which had a length above the mentioned threshold. At this point, several thousands of documents with content that is likely to be similar to some extent to the initial text T_0 are available, which are candidates to be incorporated into the ad hoc corpus pursued in the approach. This set of candidate texts are denoted as $CT(T_0)$ in Eq. (10):

$$CT(T_0) = \left\{CT_j / CT_j \text{ is the cleaned text of some } u_j \in U(T_0)\right\}. \tag{10}$$

Of course, a large number of these documents might have a tangential relationship to the initial text (e.g. a battle involving Athens corresponding to a different historical stage). It is therefore necessary to measure their similarity to the particular context, order them according to that metric, and discard the irrelevant ones.

3.6. *Assessing Relevance of Each Candidate Text*

By simple visual inspection, it is easy to notice that a significant amount of the documents obtained in the previous phase do not have a strong-enough relationship to the proposed domain, and for that reason they should not be incorporated into a domain-specific corpus. Of course, the wikicats retrieved could be quite specific (e.g. `Greco-PersianWars`) leading to documents that are likely to belong to the thematic of the initial text. But they can be also wide spectrum, mixing similar URLs with unrelated ones (e.g. `PeopleFromAthens` leading both to `Themistocles` – leader of Greeks in the scenario under consideration – and `Queen Sofia of Spain`, currently alive and probably irrelevant in the context of the battle of the Thermopylae).

To detect and discard these uninteresting documents, the similarity between the initial text T_0 and each of the candidate texts T_j is measured. Depending on these values, the relationship between T_j and T_0 can be relevant (and thus the document is incorporated into the ad-hoc corpus) or irrelevant (the document is discarded), as depicted in Eq. (11):

$$Corpus(T_0) = \{T_j / T_j \in CT(T_0) \wedge T_j \text{ is } \textit{relevant} \text{ according to } \textit{similarity}(T_j, T_0)\}. \tag{11}$$

The different *similarity* metrics that have been taken into account to detect the relationship between each of the candidate texts and T_0 are explained in Section 3.6.1. The way in which the best metric has been identified is detailed in Section 3.6.2. Finally, the *relevance* criteria considered to evaluate the candidate texts are discussed in Section 4.

[10] Beautiful Soup Python library has been used for this purpose.

3.6.1. *How to Measure Similarity Between Each Candidate T_j and T_0*

When it comes to detecting resemblance between each candidate text and the input one (T_0), part of the significant information bound to them (that has been discovered by the procedures described in previous sections) can be leveraged, such as their common wikicats and subjects. Besides, it is also possible to resort to approaches defined in the literature – based on using word-level and sentence-level embeddings – that have attained good performance for measuring semantic similarity between texts (Le and Mikolov, 2014; Lau and Baldwin, 2016; Fu *et al.*, 2018; Gali *et al.*, 2019; Dai *et al.*, 2020; Chen *et al.*, 2019). Details of the four metrics adopted in this work and the results of the experiments justifying the adoption of a Doc2Vec-based solution are presented next.

1. **Wikicats Jaccard similarity (*Sim_W*).**

   This metric measures the similarity between the initial text T_0 and the candidate one T_c according to the coincidence of the wikicats that characterise each of them. That is:

   $$WK(T_c) = \{wk_{c_i} / wk_{c_i} \text{ is a wikicat of some entity included in } DE(T_c)\}$$

   $$WK(T_0) = \{wk_{0_i} / wk_{0_i} \text{ is a wikicat of some entity included in } DE(T_0)\}$$

   $$Sim_W(T_c, T_0) = \#(WK(T_c) \cap WK(T_0)) / \#(WK(T_c) \cup WK(T_0))$$

2. **Subjects Jaccard similarity (*Sim_S*).**

   This metric is similar to the previous one but using common subjects between T_0 and T_c, instead of wikicats.

3. **spaCy similarity (*Sim_C*).**

   The computation of similarity values by means of this metric is driven by spaCy,[11] a Python package for natural language processing. As usual in similar packages, it provides functions for tokenizing texts, removing stopwords and punctuation, classifying words according grammatical categories… In addition, it also implements mechanisms to assign a vector to each word. For this task, it uses algorithms like Glove or Word2Vec (a variant of this by default) to assign a word embeddings vector to each word of the vocabulary.[12] And, naturally, it provides functions to measure similarity between words, comparing vectors through the traditional cosine-based similarity.

   Directly derived from this, spaCy also provides a simple mechanism to measure similarity between texts, generating a vector for each text (the average of the corresponding vectors for each word of the text) and computing cosine-based similarity between the text vectors.

[11] www.spacy.io

[12] Note that several English vocabularies can be loaded at startup, from small to large ones.

4. **Doc2Vec similarity (*Sim_{AP}*).**

   Some works in the literature have shown that Doc2Vec performs robustly in measuring document similarity when trained using large external corpora (Lau and Baldwin, 2016; Dai *et al.*, 2020). Bearing these results in mind, this embedding model has been explored to select the most similar candidate documents to the initial text T_0. Since the aim is to build an ad hoc corpus, at this point there was no dataset to learn a custom-trained Doc2Vec model. Therefore, a model pre-trained on some publicly available generic corpus has been adopted. In particular, this paper uses the well-known Doc2Vec model trained on a large corpus of news from the Associated Press (AP).[13]

   Using this model and the Gensim implementation of the Doc2Vec algorithm, the approach obtained the characteristic vector for both each candidate text T_c and T_0, and then used the cosine-based similarity to measure similarity between vectors (and so documents).[14] For the training process, a simple pre-processing has been carried out on every text using the Gensim API: tokenize to obtain a list of words, lowercase them, remove punctuation, and remove stop-words.[15]

3.6.2. *How to Select the Best Similarity Metric in the Approach*

In order to decide which of the four metrics described in the previous section (Sim_W, Sim_S, Sim_C and Sim_{AP}) is the best for the intended purpose, a supervised approach will be followed to measure how well they can identify the similarity between T_0 and a set of documents recognized as highly similar.

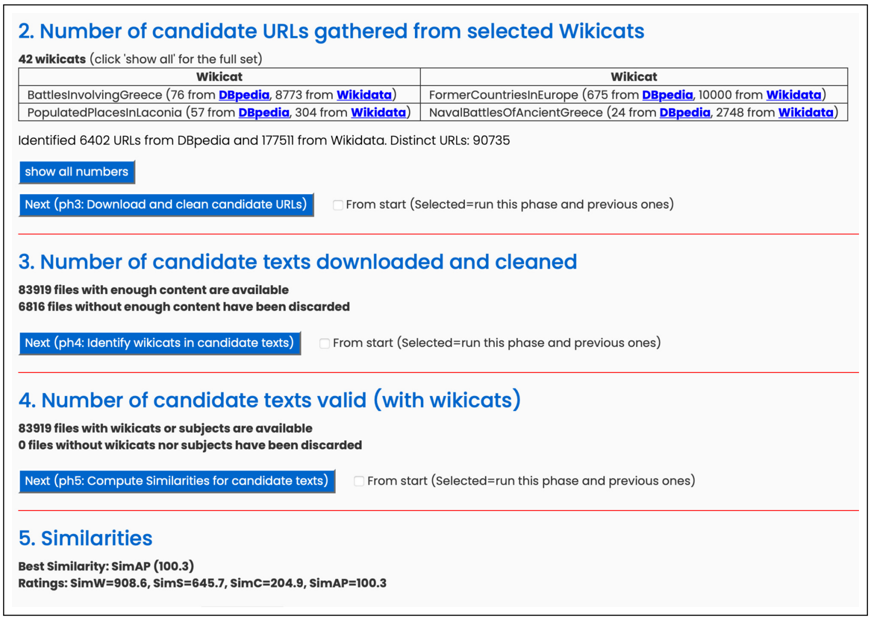
- First, the similarity between the initial text and each candidate text will be computed using each one of the similarity metrics. This allows the set of candidate texts in $CT(T_0)$ (Eq. (10)) to be sorted in decreasing order according to their similarity value Sim_x with respect to T_0, that is, each Sim_x will lead to a different $CT_x(T_0)$. In this regard, note that the starting point is the input text T_0 containing several DBpedia entities detected by the DBpedia Spotlight ($DE(T_0)$ in Eq. (2)). As such entities represent Wikipedia pages, their corresponding cleaned texts (included as candidates in $CT(T_0)$) are also available, and they will be the testing set, as it is clear that all of them are texts quite related to the subject of T_0.
- Each entity e_i appearing in T_0 will be located in a different position P_{x_i} in each $CT_x(T_0)$ (so that the higher the similarity, the lower the position). So, the best similarity metric Sim_x is the one that locates all the T_0's entities the lower the better in the set $CT_x(T_0)$.[16] In particular, the approach selects the similarity value with the lowest average position for all the entities of T_0 (denoted as $Avrg(P_x)$). The results of this procedure are shown in the snapshot of the tool developed that is depicted in Fig. 2.

[13] https://github.com/shreyanse081/gensim_Doc-Word2Vec

[14] The tutorial available at https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html provides examples on how to use GenSim to load a corpus, train a Doc2Vec model using that dataset and infer the corresponding document vector.

[15] Tests of keeping and removing stop-words before training showed that better results were obtained by removing them.

[16] Currently, the approach considers only entities of types Person, Location and Event (the 10 entities identified in the example text meet this requirement).



2. Number of candidate URLs gathered from selected Wikicats

42 wikicats (click 'show all' for the full set)

Wikicat	Wikicat
BattlesInvolvingGreece (76 from **DBpedia**, 8773 from **Wikidata**)	FormerCountriesInEurope (675 from **DBpedia**, 10000 from **Wikidata**)
PopulatedPlacesInLaconia (57 from **DBpedia**, 304 from **Wikidata**)	NavalBattlesOfAncientGreece (24 from **DBpedia**, 2748 from **Wikidata**)

Identified 6402 URLs from DBpedia and 177511 from Wikidata. Distinct URLs: 90735

show all numbers
Next (ph3: Download and clean candidate URLs)
From start (Selected=run this phase and previous ones)

3. Number of candidate texts downloaded and cleaned

83919 files with enough content are available
6816 files without enough content have been discarded

Next (ph4: Identify wikicats in candidate texts)
From start (Selected=run this phase and previous ones)

4. Number of candidate texts valid (with wikicats)

83919 files with wikicats or subjects are available
0 files without wikicats nor subjects have been discarded

Next (ph5: Compute Similarities for candidate texts)
From start (Selected=run this phase and previous ones)

5. Similarities

Best Similarity: SimAP (100.3)
Ratings: SimW=908.6, SimS=645.7, SimC=204.9, SimAP=100.3

Fig. 2. Interface of the corpus builder tool developed by the authors of the paper, which aims to evaluate the similarity between the initial text T_0 and the selected candidate texts through the four metrics considered in the approach (Sim_W, Sim_S, Sim_C and Sim_{AP}).

Returning to the example illustrated throughout this section, Table 2 shows the positions occupied by the 10 DB-SL entities (discovered in T_0) in the ordered sets that have been obtained using the four metrics considered in the tests (Sim_W, Sim_S, Sim_C and Sim_{AP}). Note that Sim_{AP} is the Doc2Vec similarity metric computed with the AP pre-trained model.

These data are not deterministic for the Doc2Vec-based similarity Sim_{AP}, because the Doc2Vec model provides slightly different similarity values for any pair of documents in every execution. This is expected behaviour, as inferring a vector for a new document is not a deterministic process but an iterative one with some randomization involved in the negative sampling feature in the training process.[17] These slight variations are not usually important when estimating a similarity, but, in the example illustrated in the paper, 83919 documents are sorted, and such variations affect significantly the final order among them (and the positions of the entities).

To solve that issue, and compare Doc2Vec with the other similarity metrics, the proposed algorithm (which is outlined in Algorithm 1) computes the results for 5 different executions, and uses the average for every entity. As depicted in Table 2, the Doc2Vec-driven metric was found to be the best one according to the defined criterion (the average

[17]Note that, opposite to Word2Vec vectors for words in the vocabulary, there is no vector stored in the model for the new documents, so such vector must be computed on the way, replicating the training process.

Table 2
Positions occupied by the 10 DB-SL entities (discovered in the initial text T_0) in a set of candidate documents that have been ordered (as per their similarity to that text) using each of the four metrics considered in the approach. The best metric is the one that finds the entities in the lowest positions in the ordered set, that is, in the documents most similar to the input text (Sim_{AP} in this case).

Entity	Sim_W	Sim_S	Sim_C	Sim_{AP}
`Battle_of_Thermopylae`	1166	624	1	1
`Battle_of_Marathon`	1049	605	11	6
`Xerxes_I`	1295	734	266	7
`Themistocles`	618	481	338	9
`Battle_of_Artemisium`	220	198	18	16
`Leonidas_I`	73	82	32	29
`Thespiae`	359	347	58	126
`Darius_I`	1133	949	430	161
`Sparta`	3172	2422	613	265
`Ephialtes_of_Trachis`	1	15	282	383
Average	908.6	645.7	204.9	100.3

is 100.3). In light of the results, the Doc2Vec similarity metric (denoted by Sim_{AP}) is adopted to order the candidate texts in $CT(T_0)$, hereafter denoted as $CT_{AP}(T_0)$.

Algorithm 1. Sketch of the ad hoc corpus building algorithm.

```
1. Provide the input text ($T_0$) to contextualize the application domain.
2. Compute $DE(T_0) \rightarrow$ DB-SL entities identified in $T_0$ (Eq. (2)).
3. Compute $WK(T_0) \rightarrow$ Wikicats that characterise $T_0$ (Eq. (5)).
4. Compute $U(T_0) \rightarrow$ URLs associated to the Wikicats of $T_0$ (Eq. (10)).
5. Compute $CT(T_0) \rightarrow$ Clean texts (candidates) obtained from the previous URLs (Eq. (11)).
6. Get $Corpus(T_0)$ from $CT_{AP}(T_0) \rightarrow$ Subset of texts in $CT(T_0)$ with the highest $Sim_{AP}$ similarity with $T_0$
   (selected by the experiments described in Sections 3.6.2 and 4.2.2).
```

As depicted in above sketched algorithm, a subset of the candidate documents that occupy the top positions in $CT_{AP}(T_0)$ should be selected to be incorporated into the ad hoc corpus. As can be guessed, there is no red line marking which candidates should be included in the corpus and which should not. There should be negligible differences among candidates on either side of any threshold. Of course, the more candidates are added, the better it is to achieve a more stable embeddings model, but the documents will be less and less similar to the initial text. And it also seems clear that, at the end, any consistency measure can be influenced by the final application in which the corpus is used.

But one thing that can be done is to analyse different scenarios and study their results according to a common criterion. This allows to assess the performance of the models obtained by training on different subsets of the $CT_{AP}(T_0)$, and to observe how the selected size affects them. This is the criterion adopted in the next section to validate the proposed approach to select the documents that will finally be incorporated into the tailor-made corpus.

4. Validating the Selection of In-Domain Documents for the Corpus

The starting point is the collection of candidate documents that have been gathered from Wikipedia and ordered according to their similarity to T_0. Recall that such ordered set was denoted as $CT_{AP}(T_0)$ as it was the result of using the best metric ($SimAP$), which is based on the pre-trained Doc2Vec model on the AP news collection. The documents in this set are ordered from most to least similar to T_0, but it is likely that not all of them are strongly-enough related to the context because they have been collected from some wikicats that may be tangentially related to the initial text. Therefore, only a subset of these documents needs to be considered. In particular, only the texts that occupy the first positions of the ordered set should be incorporated into the ad hoc corpus being pursued. As there are no clear guidelines on this threshold (the similarity results are certainly quite continuous over the 83919 computed values), different thresholds have been chosen in order to be able to use the resulting corpora in some scenario and to evaluate the quality of the results obtained. In this regard, it was decided to select all the percentages of the set $CT_{AP}(T_0)$, from 1% to 10%, in order to train the corresponding Doc2Vec models with that set of ad hoc corpora (resulting in sizes 839, 1678, $\ldots$). The consistency of such models (denoted as M_1 to M_{10}) was then analysed individually (Section 4.1) and compared with each other and with the generic AP model (Section 4.2).

This comparison should serve to answer a relevant outstanding question: "*are any of these ad hoc models better than the generic AP model?*' Note that this is the cornerstone of the research described in the paper, which aims to devise a procedure to build an ad hoc corpus (composed of documents significantly related to a given application scenario), as there are several references in the literature stating that the model derived from such a corpus should be better than a model trained on a generic corpus of documents (not particularly related to the scenario under consideration). Therefore, evidences in this respect should be provided.

4.1. *Consistency Tests for Evaluating Embedding Models Learned from Ad Hoc Corpora*

One simple consistency test to check the resulting models is to compute the self-rank of each training document when searching for similar documents. That is, for each one of the training files, the model is asked for the N most similar documents to it, as if such file were new and not already in the model. Obviously, the model should select the same file as the most similar to itself (1-*rank*), that is, the first in the list of most similar docs.[18] But sometimes, the model makes a mistake and finds other document that is even more similar. The less mistakes, the better the model.

Table 3 depicts the results for the 10 Doc2Vec models (which have been trained on the 10 ad hoc corpora), where the $Rank_i$ row shows the percentage of training document that had a 1-$rank$ for each M_i model. The rather high value (close to 1) of the resulting figures confirms that the behaviour of all learned ad hoc models is consistent.

[18] See https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html for details.

Table 3
1-*rank* results obtained for 10 ad hoc Doc2Vec models generated in the approach (M_1 to M_{10}). The results confirm that, given a training document, these models were almost always correct in identifying that document as the most similar to itself (on average, this was true for 98.5% of the training documents).

M_i	1	2	3	4	5	6	7	8	9	10
$Rank_i$	94.5	98.9	99.2	99	98.9	98.8	98.8	99	98.9	98.8

Table 4
Similarity values that our 10 Doc2Vec models have measured between documents that are related to T_0 to different extents. The results confirm that the 10 ad hoc models are able to detect high similarity between strongly related documents (values close to 1 included in the rows corresponding to the average of S_1 and S_2), and low similarity between unrelated texts (very low values referring to the average of S_3).

M_i	1	2	3	4	5	6	7	8	9	10
Avg S_1	0.93	0.93	0.93	0.93	0.94	0.93	0.94	0.94	0.94	0.93
Avg S_2	0.83	0.84	0.83	0.84	0.84	0.83	0.84	0.84	0.84	0.85
Avg S_3	0.18	0.19	0.19	0.19	0.18	0.21	0.21	0.18	0.19	0.22

Another simple consistency check for the models consists of observing how well they discriminate between similar and dissimilar documents. To this aim, a simple experiment was made where two lists of documents in $CT_{AP}(T_0)$ were selected:

- L_1: the 100 most similar documents to the initial text, according to the Sim_{AP} similarity metric.
- L_2: the 100 less similar documents to the initial text, as per the same metric.[19]

Each of these 200 documents was divided into two parts of equal size, and several similarity values were computed for each one of the 10 ad hoc Doc2Vec models:

- S_1: similarity values between both parts of each file in L_1 (**these values should be high** as the document is similar to T_0 and both parts are about the same theme). A list of 100 similarity values was obtained for each model.
- S_2: similarity values between both parts of each file in L_2 (**they should be also high**, as both parts are about the same theme). A list of 100 values was computed for each model.
- S_3: similarity values between the first part of each file in L_1 and the first part of the file occupying the same index in L_2 (**these values should be low** as most of the time they are unrelated documents, as can be seen from the example mentioned earlier on `Themistocles` and `Queen Sofia of Spain`).

The average of the similarities S_1 to S_3 was finally computed for each model M_i, as depicted in Table 4. Once again, the results confirm that the 10 ad hoc models are completely consistent with what expected (similarities between both parts of similar or dissimilar documents are extremely high while cross similarities are low).

[19] Actually, only documents larger than 3KB were included in each of the lists because very small documents were of little significance for testing purposes.

4.2. *Comparison to the Doc2Vec-AP Model*

The previous tests showed that the models generated in this approach work well, but they did not confirm which one is the best or if they are better than the generic Doc2Vec AP model. To try to shed light on such issue, the performance of the models when they are used in some scenario has to be evaluated. The methodology adopted in the validation and the discussion on results obtained are detailed in Section 4.2.1 and Section 4.2.2, respectively.

4.2.1. *Experimental Methodology*

The proposed validation scenario is inspired by the procedure that was adopted to select the similarity metric based on the Doc2Vec AP model (Sim_{AP}) as the best one among the 4 possibilities explored in Section 3.6.2. In particular, the procedure is organized as follows:

1. First, the M_1 to M_{10} models were used to measure the Doc2Vec similarity among T_0 and each candidate text (included in the set $CT_{AP}(T_0)$). This led to 10 sets whose documents were arranged in decreasing order according to the similarity values measured with these models (denoted as $CT_1(T_0)$ to $CT_{10}(T_0)$).
2. Next, the focus was put on the positions of the T_0's entities in the ordered sets. These position values were averaged to obtain a consistency indicator that allowed the 10 ad hoc models to be compared with each other and with the generic *AP* one.
3. For a more robust comparison in the application scenario linked to the *Battle of Thermopylae*, the list of 10 DB-SL entities initially discovered was extended with new entities that were actually significant in the context of the second Persian invasion of Greece (but were not mentioned in the initial text T_0), as depicted in Table 5. This way, a very descriptive set of 18 DBpedia entities that characterised the context under consideration was defined. Of course, all of them were included in the set of candidates

Table 5
Set of 18 DB-SL entities considered in the experimental validation in the context of the Battle of Thermopylae.

The 10 DB-SL entities initially identified from the initial text T_0		
Events	People	Places
`Battle_of_Thermopylae`	`Leonidas_I`	`Thespiae`
`Battle_of_Artemisium`	`Xerxes_I`	`Sparta`
`Battle_of_Marathon`	`Themistocles`	
	`Ephialtes_of_Trachis`	
	`Darius_I`	
The new 8 DB-SL entities about the Battle of Thermopylae which are not present in T_0		
Events	People	
`Battle_of_Salamis`	`Mardonius`	
`Battle_of_Mycale`	`Hydarnes`	
`Battle_of_Plataea`	`Hydarnes_II`	
	`Immortals_(Achaemenid_Empire)`	
	`Herodotus`	

Table 6
Positions occupied by the 18 DB-SL entities of T_0 within a set of candidate texts that have been ordered (as per their similarity with T_0) considering the generic AP model and our 10 in-domain Doc2Vec models.

DB-SL entity	*AP*	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
`Battle_of_Thermopylae`	1	1	5	6	5	7	3	4	5	6	9
`Battle_of_Plataea`	3	6	2	2	1	1	14	1	1	2	5
`Battle_of_Salamis`	4	9	8	9	11	9	11	18	12	10	13
`Battle_of_Marathon`	6	5	3	1	2	6	7	6	14	22	11
`Themistocles`	7	46	49	71	58	32	76	40	42	72	68
`Xerxes_I`	9	19	20	107	55	54	68	37	58	65	56
`Battle_of_Mycale`	15	2	1	3	3	2	2	2	2	1	2
`Leonidas_I`	16	3	9	20	26	17	19	16	27	35	34
`Battle_of_Artemisium`	20	15	10	8	10	8	10	12	8	9	10
`Hydarnes_II`	75	26	37	13	19	64	60	53	87	57	69
`Darius_I`	139	64	90	164	83	106	186	171	140	136	125
`Mardonius`	142	4	12	11	8	14	12	20	34	18	31
`Thespiae`	189	731	382	150	289	193	50	110	124	168	70
`Sparta`	246	680	399	443	438	393	547	491	302	903	765
`Ephialtes_of_Trachis`	297	33	33	54	38	55	55	120	69	96	94
`Inmortals_(Achaemenid_Empire)`	1483	392	303	236	311	369	404	644	487	464	607
`Hydarnes`	1859	95	77	101	69	72	77	84	103	141	166
`Herodotus`	9287	282	173	1349	606	799	748	694	683	892	774

collected in the initial phases described in Section 3. Moreover, their similarity to T_0 could be calculated with the Doc2Vec algorithm using the testing models M_1 to M_{10}, and they were also present in the aforementioned ordered sets $CT_x(T_0)$ calculated from these models (with $x \in [1, 10]$).

4. Lastly, the positions of those 18 DBpedia entities were searched in the sets $CT_1(T_0)$ to $CT_{10}(T_0)$. A typical run (remember that results slightly change from run to run due to the randomness features of the Doc2Vec algorithm) of the results obtained with both the 10 ad hoc models and the AP generic one is depicted in Table 6.

4.2.2. *Discussion on Experimental Results*

As shown in Table 6, most of the entities occupy relevant positions in all of the studied orderings. This happened to be always true, even though the actual numbers change among different trainings of the same subset of candidates. As mentioned before, the Doc2Vec training algorithm involves some randomness in its steps (for instance, some data is randomly discarded to accelerate the convergence without having significant influence in the final results). This is not important for calculating a given similarity, as the fact that this value is 0.865 or 0.863 should not make any difference. But when ordering 83919 documents, these little differences can lead to all entities slightly changing their positions up or down. For instance, the `Battle of Plataea` (the final battle of the Second Persian Invasion of Greece) moves between position 1 and position 6, all of them quite important rankings when ordering 83919 documents, in any case.

In addition, some discordant values have been highlighted with background gray color in Table 6. They are clearly outliers that have to be taken into account to conduct a more appropriate analysis. In this regard, note that word embeddings are machine learning techniques that can be influenced by many factors that lead sometimes to unexpected results. For example, it may happen that a key figure of the historical event under consideration is described in the candidate text (retrieved from Wikipedia) in a way that is not rich enough for these methods to lead to the expected similarity values. This is the case for the candidate text of the entity `Herodotus`: although he is the main source of information about the `Battle of Thermopylae` (hence his mention in the candidate document), in reality this battle is only a small part of his work as a historian. Experiments have also confirmed the influence of other aspects. In particular, even considering the same set of texts, there are several training parameters that can greatly affect the resulting model. With different training sets, these differences can be even more pronounced.

For the above reasons, it is natural that outliers appear. These are documents that should have been rated high, but are not. This happens both with small ad hoc corpora and with the huge generic AP corpus. But, as shown in Table 6, these outliers are not the same in all cases (although they are quite similar in the ad hoc corpora). Because of this, in order to appropriately compare ad hoc models among them and with AP model, it is convenient to remove such outliers, as the overall quality of a model should be assessed by the most of its results, and not influenced by a small number of irregular items.

This way, the Z-score (Jiang *et al.*, 2009; Aggarwal, 2017) and the IQR (Tukey, 1977; Sunitha *et al.*, 2014) methods were adopted to identify outliers. Both methods confirmed the highlighted values as discordant with the previous ones. Only values over position 400 were analysed as it is clear that low values may be discordant from a mathematical point of view but still significative regarding similarity.[20]

When removing discordant outliers, it should be borne in mind that averages are being compared and therefore it is necessary to include the same number of elements in the calculation. This is due to the fact that outliers occupy the highest positions in any ranking and simply discarding them in their individual rankings would lead to results that are not directly comparable. Thus, the different rankings were studied to find the one with the highest number of outliers (3 in AP), and this amount was removed from all models. Thus, the 15 entities ranked in the lowest positions were considered for all models. It is worth noting that this is the best case scenario for the AP model (the 3 highest entities were removed from that ordering, thus reducing its mean and giving more value to any other model that outperforms AP).

The results obtained with the 10 ad hoc models learned by taking as training datasets different percentages (from 1% to 10%) of the candidate texts in $CT(T_0)$ are shown in the top row of Table 7, together with the outcome of the existing AP Doc2Vec model. Note that the results include the average of several runs, considering only the entities found in the 15 lowest positions for each model.

[20]For example, in the series 1, 2, 3, 10 the last one would be an outlier from a pure mathematical point of view, but it is obvious that, when talking about positions within a set of 83919 elements, it is a low position that must be understood as a relevant value regarding any similarity ranking.

Table 7
Average positions occupied by the 15 best ranked entities, considering both the AP model and 20 Doc2Vec in-domain models learned by taking between 1% and 20% of the initially-selected candidate documents. The results confirm that the M_5 model obtains the best outcome as it allows finding the 15 entities in the candidate documents that have been selected as the most similar to the input text (i.e. those occupying the lowest positions in the set $CT_5(T_0)$).

M_i	AP	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
$Avrg_i$	**78**	58	46	52	47	**43**	44	46	55	58	**83**

M_i	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}	M_{17}	M_{18}	M_{19}	M_{20}
$Avrg_i$	68	60	61	74	66	75	**93**	**105**	**92**	**90**

Based on the results, when compared to the 78-score[21] of the *AP* model, the ad hoc models show excellent average positions for percentages ranging from 1% to 10%. This outcome is reasonable because when training with a small percentage of candidate documents, the selected texts are on the topic of the initial one (i.e. they are very similar to T_0). Consequently, the models learned from these ad hoc corpora are effective at measuring similarities between T_0 and any of these documents. Particularly, as shown in Table 7, the ad hoc models M_1 to M_9 outperform the generic *AP* model. This can be observed from the consistently lower average positions measured by each of these models, which are always significantly below 78. Notably, among them, model M_5 achieves the best results.

To examine the evolution of the consistency across a larger number of models derived from different percentages of $CT(T_0)$, we replicated the above procedure for values ranging from 11% to 20%, as shown in the bottom row of Table 7. The results indicate that the quality of the learned ad hoc models starts to degrade as an increasing number of documents are included in the training. This weakening is attributed to their reduced relation to the thematic of T_0, leading to the resulting model being less focused on the target scenario.

Based on Table 7, it can be observed that models trained with high percentages (17% to 20%) of candidate documents perform less effectively than the generic model *AP*. The inclusion of a large number of out-of-domain training texts leads to the development of generic models. Consequently, models M_{17} to M_{20} are inferior to the *AP* model, as the latter has been trained with a more extensive document set, making it better suited for detecting similarities between two generic texts rather than being specifically tailored to the topic of T_0.

To sum up, the results presented in Table 7 confirm that the initial ad hoc models significantly outperform the generic AP model. For the domain studied in this paper, a corpus containing only 5% of the candidate texts yielded the best results with an average position of 43, as opposed to the AP model's 78-score. This implies that, based on the criteria employed in this study, such ad hoc models (which have been constructed fully automatically, without requiring human contributions except for optimizing corpus creation) are more effective than the generic AP model in identifying the documents most relevant to a specific context defined in the initial text T_0.

[21] Recall that this score means that 78 is the average of the positions occupied by the 15 DB-SL entities in the ordered set of candidate texts. The lower the average, the better, as this means that the relevant entities have been identified by the model as very similar to the initial text.

5. Conclusions and Further Work

In this paper, an automatic procedure based on the Linked Open Data infrastructure has been proposed, which allows easily to obtain ad hoc corpora (from a user-specified short input text) that bring benefits for existing word-level and document-level embedding models. So far, such models have been fine-tuned on small collections of in-domain documents in order to improve the performance. These documents are often compiled manually and without assessing in any way the relevance of each text in the particular domain. In opposite, the approach described in this paper automatically gathers numerous in-domain training texts by relying on NER tools and state-of-the-art embedding models in order to guarantee a meaningful relationship between each possible training document and the initial text.

On the one hand, DBpedia-Spotlight is used to recognize DBpedia named entities in the initial text and drive the process of building an initial ad hoc corpus. On the other one, Doc2Vec models allow to identify new relevant in-domain texts (to be incorporated into the ad hoc training dataset). This way, the final tailor-made corpus brings together a large amount of meaningful and precise information sources, which lead to learning high quality domain-specific embeddings.

These dense vector representations accurately model domain peculiarities, which is especially critical for exploiting the language representation capabilities of embedding models in very particular fields (e.g. medicine, History or mechanical engineering, just to name a few). These fields are not conveniently considered in either the huge publicly available generic training datasets or the small-sized hand-collected domain datasets adopted in some existing approaches. Unlike these works, our approach is able to build, without human assistance, a training corpus on any topic and domain that can be exploited by existing models, requiring only to provide as input a variable-length piece of text.

In line with this, well-known models (like Word2Vec and GloVe) could take advantage of this approach to fight the Out-Of-Vocabulary problem, which is stemmed from the usage of generic training corpora. This limitation has been traditionally alleviated in other models (like FastText) by working at subword-level, which introduces excessive computational costs and memory requirements (Armand *et al.*, 2017). However, this approach enables to embed in-domain words that would be rare/unusual in a publicly-available generic corpus, and therefore impossible to be learned from general-domain datasets or even from imprecise/incomplete domain-specific datasets.

Our procedure to custom corpus construction shows several advantages over the different approaches presented in the related work section that address similar objectives: the approach here described is more autonomous, less dependent on user feedback to guaranty the quality of outputs; in our work, the relevance of the documents for the given subject is measured before including them in the corpus; and the search process is based on an open large repository where new unknown documents can be discovered, analysed and included into the corpus, besides being used to trigger new searching processes, thus leading to larger custom corpora composed of thousands of documents.

As an honest limitation, the paper concentrates on evaluating the performance of the proposed method using Doc2Vec embeddings, while not considering other Transformer-based models. Despite acknowledging the potential and remarkable results achieved by sophisticated Transformer architectures like BERT, such approaches have been omitted due to certain characteristics of the available models that do not align well with the specific objectives of our validation. In particular, our experimental validation has demonstrated that the performance of the custom-built in-domain corpus, when compared to a generic training dataset, is superior within the context of the specific embedding model used (Doc2Vec in this case). To achieve this, we trained a Doc2Vec model from scratch using the tailored collection and compared it to a generic model. However, training a BERT model from scratch in the case of Transformers is not feasible for us due to its unaffordable computational requirements. Instead, it is common practice to start with a pre-trained model on a massive collection of generic information (referred to as base model) and then fine-tune it for specific NLP tasks and custom corpora.

Given that our approach aimed to compare a model trained from scratch on the adhoc collection against one trained on a generic collection, we found the use of Doc2Vec more suitable for the purposes of our research. This choice was driven by the need for a more equitable comparison scenario between the general and ad hoc approaches, which is made possible by comparing models trained from scratch with Doc2Vec. The proposed experimental validation has tested the research hypothesis considered in the approach and has demonstrated that the performance of the automatically-built in-domain corpus is better than that of a generic training dataset in the context of a particular embedding model (Doc2Vec in this case). In reality, replicating or even improving this behaviour with other recent and sophisticated Transformer-based embedding models (such as GPT, BERT and their multiple variants) does not invalidate the results obtained in this work with Doc2Vec. In particular, apart from the aforementioned reasons, Doc2Vec presents two additional compelling advantages. Firstly, its good performance against related document-level models (Bhattacharya *et al.*, 2022; Kim *et al.*, 2018; Grefenstette *et al.*, 2013; Mikolov *et al.*, 2013a; Kiros *et al.*, 2015, 2018) and secondly, the existence of a mature implementation of it through GenSim (Rehürek and Sojka, 2010), which allowed to train own models from scratch and evaluate the effects of the training corpus on the quality of the resulting models (rather than being able to simply use pre-trained models that have been learned from inaccessible documents).

Regarding the further work, having experimentally validated that in-domain corpora improve generic training datasets in a very specific domain, short-term research plans to explore the performance of models that have first been learned from a generic corpus and then fine-tuned on a collection of in-domain texts (which will be automatically retrieved by the proposed algorithm). The goal of these experiments is to incorporate a diverse array of models, encompassing advanced Transformer-based approaches like BERT, along with numerous other models that are continually emerging in the literature.

A. Mathematical Notation Adopted

Notation	Meaning	Equations
$DE(T_0)$	Set of DBpedia entities that DBpedia Spotlight identifies in the input text T_0, which share common Wikicats and/or subjects.	(1) and (2)
$WK(e)$	Set of relevant Wikicats that characterise a given entity e.	(3) and (4)
$WK(T_0)$	Set of relevant Wikicats that characterise the input text T_0.	(5)
$U_{DB}(w_k)$	Set of URLs of pages dealing with entities tagged with the w_k wikicat in DBpedia.	(6)
$U_{WK}(w_k)$	Set of URLs of pages dealing with entities tagged with the w_k wikicat in Wikidata.	(7)
$U_{DW}(w_k)$	Union of the sets $U_{DB}(w_k)$ and $U_{WK}(w_k)$.	(8)
$U(T_0)$	Set of URLs that are associated with some wikicat included in $WK(T_0)$.	(9)
$CT(T_0)$	Set of documents, related to T_0 in some extent, that are candidates to be incorporated into the ad hoc training corpus.	(10)
$Corpus(T_0)$	Custom-built training corpus that includes only the domain-specific documents in $CT(T_0)$ that are significantly related to T_0.	(11)
$Sim_W(T_c, T_0)$	Semantic similarity metric based on the common wikicats identified between the candidate text T_c and the input text T_0.	3.6.1 and 3.6.2
$Sim_S(T_c, T_0)$	Semantic similarity metric based on the common subjects identified between T_c and t T_0.	3.6.1 and 3.6.2
$Sim_C(T_c, T_0)$	Semantic similarity metric between T_c and T_0 measured by the spaCy Python package.	3.6.1 and 3.6.2
$Sim_{AP}(T_c, T_0)$	Semantic similarity metric between T_c and T_0 measured by the existing AP Doc2Vec model (which has been trained with a generic collection of Associated Press news).	3.6.1 and 3.6.2
$CT_W(T_0)$	Set resulting from sorting $CT(T_0)$ in decreasing order as per the similarity values measured (between each T_c and T_0) by the metric $Sim_W(T_c, T_0)$.	3.6.2
$CT_S(T_0)$	Set resulting from sorting $CT(T_0)$ in decreasing order as per the similarity values measured (between each T_c and T_0) by the metric $Sim_S(T_c, T_0)$.	3.6.2
$CT_C(T_0)$	Set resulting from sorting $CT(T_0)$ in decreasing order as per the similarity values measured (between each T_c and T_0) by the metric $Sim_C(T_c, T_0)$.	3.6.2
$CT_{AP}(T_0)$	Set resulting from sorting $CT(T_0)$ in decreasing order as per the similarity values measured (between each T_c and T_0) by the metric $Sim_{AP}(T_c, T_0)$.	3.6.2
M_i with $i \in [1, 20]$	Doc2Vec model learned from an ad hoc training corpus including i% (from 1% in M_1 to 20% in M_{20}) of the candidate documents in $CT_{AP}(T_0)$.	4.1 and 4.2
$CT_i(T_0)$, $i \in [1, 20]$	Set resulting from sorting $CT_{AP}(T_0)$ by the Doc2Vec ad hoc model M_i.	4.2.1 and 4.2.2

References

Abacha, A., Dina, D. (2016). Recognizing question entailment for medical question answering. *AMIA Annual Symposium Proceedings*, 2016, 310–318.

Adjali, R., Besancon, R., Ferret, O., LeBorgne, H., B., G. (2020). Multimodal entity linking for tweets. In: *Proceedings of the 42th European Conference on Advanced Information Retrieval*, Lisbon, Portugal.

Aggarwal, C.C. (2017). *Outlier Analysis*. Springer, New York.

An, Y., Liu, S., Wang, H. (2020). Error detection in a large-scale lexical taxonomy. *Information*, 11(2). https://doi.org/10.3390/info11020097.

Armand, J., Grave, E., Bojanowski, P., Mikolov, T. (2017). Bag of tricks for efficient text classification. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics (ACL), Valencia, Spain, pp. 427–431. https://www.aclweb.org/anthology/E17-2068.

Axelrod, A., He, X., Gao, J. (2011). A domain adaptation via pseudo in-domain data selection. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics (ACL), Edinburgh, Scotland, pp. 355–362. https://www.aclweb.org/anthology/D11-1033.

Bansal, B., Srivastava, S. (2019). Hybrid attribute based sentiment classification of online reviews for consumer intelligence. *Applied Intelligence*, 49, 137–149. https://doi.org/10.1007/s10489-018-1299-7.

Barbaresi, A. (2013a). Challenges in web corpus construction for low-resource languages in a post-BootCaT world. In: *Proceedings of the 6th Human Languages Technologies as a Challenge for Computer Science and Linguistics*, Poznan, Poland.

Barbaresi, A. (2013b). Crawling microblogging services to gather language-classified URLs workflow and case study. In: *Proceedings of the ACL Student Research Workshop*, Sofia, Bulgaria.

Barbaresi, A. (2014). Finding viable seed URLs for web corpora: a scouting approach and comparative study of available resources. In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, Gothenburg, Sweeden.

Baroni, M., Bernardini, S. (2004). BootCaT: bootstrapping corpora and terms from the web. In: *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, Lisbon, Portugal.

Baroni, M., Kilgafrriff, A., Pomikalk, J., Rychly, P. (2006). WebBootCaT: a web tool for instant corpora. In: *Proceedings of the 12th EURALEX International Congress*, Torino, Italy.

Barrena, A., Soroa, A., Agirre, E. (2015). Combining mention context and hyperlinks from wikipedia for named entity disambiguation. In: *Proceedings of the 4th Joint Conference on Lexical and Computational Semantics*, Denver, USA.

Bhattacharya, P., Ghosh, K., Pal, A., Ghosh, S. (2022). Legal case document similarity: You need both network and text. *Information Processing & Management*, 59(6). https://doi.org/10.1016/j.ipm.2022.103069.

Blanco, Y., Gil-Solla, A., Pazos-Arias, J.J., Ramos-Cabrer, M., Daif, A., López-Nores, M. (2020). Distracting users as per their knowledge: combining linked open data and word embeddings to enhance history learning. *Expert Systems with Applications*, 143, 1–16. https://doi.org/10.1016/j.eswa.2019.113051.

Bollegala, D., Maehara, T., Kawarabayashi, K. (2015). Learning word representations from relational graphs. In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. AAAI Press, Austin Texas, USA, pp. 2146–2152. https://arxiv.org/pdf/1412.2378.pdf.

Cano, E., Morisio, M. (2017). Quality of word embeddings on sentiment analysis tasks. In: *Proceedings of the 22nd International Conference on Natural Language & Information Systems*, Liege, Belgium, pp. 1–8. https://doi.org/10.1007/978-3-319-59569-6_42.

Castagnoli, S. (2015). Using the Web as asource of LSP corpora in the terminology classroom. In: *Wacky! Working Papers on the Web as Corpus*.

Chen, Q., Peng, Y., Lu, Z. (2019). BioSentVec: creating sentence embeddings for biomedical texts. In: *Proceedings of the IEEE International Conference on Healthcare Informatics*. Association for Computational Linguistics (ACL), Xian, China, pp. 1–5. https://arxiv.org/abs/1810.09302.

Chiu, B., Crichton, G., Korhonen, A., Pyysalo, S. (2016). How to train good word embeddings for biomedical NLP. In: *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*. Association for Computational Linguistics (ACL), Berlin, Germany, pp. 166–174. https://www.aclweb.org/anthology/W16-2922.

Crystal, D. (2011). *Internet Linguistics*. Routledge, London.

Dai, A.M., Olah, C., Le, Q.V. (2020). Document embedding with Paragraph Vectors. https://arxiv.org/abs/1507.07998.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K. (2019). Bert: pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics (ACL), Minneapolis, USA, pp. 4171–4186. https://doi.org/10.18653/v1/N19-1423.

Ethayarajh, K. (2019). How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics (ACL), Hong Kong, China, pp. 55–65. https://arxiv.org/abs/1909.00512v1.

Faustini, P.H.A., Covões, T.F. (2017). Fake news detection in multiple platform and languages. *Expert Systems with Applications*, 158, 1–9. https://doi.org/10.1016/j.eswa.2020.113503.

Fu, M., Qu, H., Huang, L., Lu, L. (2018). Bag of meta-words: a novel method to represent document for the sentiment classification. *Expert Systems with Applications*, 113, 33–43. https://doi.org/10.1016/j.eswa.2018.06.052.

Gali, N., Mariescu-Istodor, R., Hostettler, D., Franti, P. (2019). Framework for syntactic string similarity measures. *Expert Systems with Applications*, 129(1), 169–185. https://doi.org/10.1016/j.eswa.2019.03.048.

Ganea, E., Hofmann, T. (2017). Deep joint entity disambiguation with local neural attention. In: *Proceedings of the 17th Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark.

Gatto, M. (2014). *Web as Corpus: Theory and Practice*. A&C Black, London.

Geng, Z., Zhang, Y., Han, Y. (2021). Joint entity and relation extraction model based on rich semantics. *Neurocomputing*, 429. https://doi.org/10.1016/j.neucom.2020.12.037.

Giatsoglou, M., Vozalis, M.G., Diamantaras, K., Chatzisavvas, K. (2017). Sentiment analysis leveraging emotions and word embeddings. *Expert Systems with Applications*, 69(1), 214–224. https://doi.org/10.1016/j.eswa.2016.10.043.

Grefenstette, E., Dinu, G., Zhang, Y., Sadrzadeh, M., Baroni, M. (2013). Multi-step regression learning for compositional distributional semantics. In: *Proceedings of Conference on Empirical Methods in Natural Language Processing*, Postdam, Germany, pp. 131–142. https://www.aclweb.org/anthology/W13-0112.

Gu, Y., Tinn, R., Cheng, H., Lucas, M., Usuyama, N., Liu, X., Naumann, T., Gao, J., Poon, H. (2021). Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare*, 3(1). https://doi.org/10.1145/3458754.

Gutiérrez-Batista, K., Campaña, J.R., Vila, M.A., Martin-Bautista, M. (2018). An ontology-based framework for automatic topic detection in multilingual environments. *International Journal of Intelligent Systems*, 33, 1459–1475. https://doi.org/10.1002/int.21986.

Han, L., Kashyap, A.L., Finin, T., Mayfield, J., Weese, J. (2013). UMBC_EBIQUITY-CORE: semantic textual similarity systems. In: *Proceedings of the 2nd Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, Atlanta, USA, pp. 44–52. https://aclanthology.org/S13-1005.

Hu, L., Ding, J., Shi, C., Shao, C., Li, S. (2020). Graph neural entity disambiguation. *Knowledge-Based Systems*, 195. https://doi.org/10.1016/j.knosys.2020.105620.

Ismayilov, A., Kontokostas, D., Auer, S., Lehmann, J., Hellmann, S. (2015). Wikidata through the eyes of DBpedia. *Semantic Web*, 9(4), 1–11. https://doi.org/10.3233/SW-170277.

Jiang, F., Sui, Y., Cao, C. (2009). Some issues about outlier detection in rough set theory. *Expert Systems with Applications*, 36(3), 4680–4687. https://doi.org/10.1016/j.eswa.2008.06.019.

Jung, G., Shing, J., Lee, S. (2022). Impact of preprocessing and word embeddings on extreme multi-label patent classification tasks. *Applied Intelligence*, 3(), 4047–4062. https://doi.org/10.1007/s10489-022-03655-5.

Khatua, A., Khatua, A., Cambria, E. (2019). A tale of two epidemics: Contextual Word2Vec for classifying Twitter streams during outbreaks. *Information Processing & Management*, 56(1), 247–257. https://doi.org/10.1016/j.ipm.2018.10.010.

Kilgarriff, A., Reddy, S., Pomikalek, J., Avinesh, P. (2014). A corpus factory for many languages. In: *Proceedings of the 7th International Conference on Language Resources and Evaluation*, Malta.

Kim, D., Seo, D., Cho, S., Kang, P. (2018). Multi-co-training for document classification using various document representations: TFIDF, LDA, and Doc2Vec. *Information Sciences*, 477, 15–29. https://doi.org/10.1016/j.ins.2018.10.006.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R.S., Torralba, A., Urtasun, R., Fidler, S. (2015). Skip-Thought vectors. In: *Proceedings of the Neural Information Processing Systems Conference*, pp. 1–11. http://arxiv.org/abs/1506.06726.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R.S., Torralba, A., Urtasun, R., Fidler, S. (2018). An efficient framework for learning sentence representations. In: *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada. https://arxiv.org/abs/1803.02893.

Lamsiyah, S., Mahdaouy, A., Espinasse, B., Alaoui, S. (2021). An unsupervised method for extractive multi-document summarization based on centroid approach and sentence embeddings. *Expert Systems with Applications*, 167(1), 1–16. https://doi.org/10.1016/j.eswa.2020.114152.

Lastra, J.J., Goikoetxea, J., Mohamed, A., Garcia-Serrano, A., Mohamed, B., Agirre, E. (2019). A reproducible survey on word embeddings and ontology-based methods for word similarity: linear combinations outperform the state of the art. *Engineering Applications of Artificial Intelligence*, 85, 645–665. https://doi.org/10.1016/j.engappai.2019.07.010.

Lau, J.H., Baldwin, T. (2016). An empirical evaluation of Doc2Vec with practical insights into document embedding Generation. In: *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics (ACL), Berlin, Germany, pp. 78–86. https://www.aclweb.org/anthology/W16-1609.

Le, Q., Mikolov, T. (2014). Distributed representations of sentences and documents. In: *Proceedings of the 31st International Conference on Machine Learning (ICML)*. Association for Computational Linguistics (ACL), Beijing, China, pp. 1188–1196. https://arxiv.org/abs/1405.4053.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Bizer, C. (2012). DBpedia: a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 9(4), 1–5. https://doi.org/10.3233/SW-140134.

Liu, X., Gao, J., He, X., Deng, L., Duh, K., Wang, Y.Y. (2015). Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics (ACL), Denver, USA, pp. 912–921. https://www.aclweb.org/anthology/N15-1092.

Logeswaran, L., Lee, H. (2018). An efficient framework for learning sentence representations. In: *Proceedings of the 6th International Conference on Learning Representations*. Association for Computational Linguistics (ACL), Vancouver, Canada, pp. 1–16. https://arxiv.org/abs/1803.02893v1.

Lynn, T., Scannell, K., Maguire, E. (2015). Minority language twitter: part-of-speech tagging and analysis of Irish tweets. In: *Proceedings of Workshop on Noisy User-generated Text*, Beijing, China. https://doi.org/10.18653/v1/W15-4301.

Ma, J., Duanyang, L., Yonggang, C., Haodong, Z., Xhang, X. (2021). A knowledge graph entity disambiguation method based on entity-relationship embedding and graph structure embedding. *Computational Intelligence and Neuroscience*. https://doi.org/10.1155/2021/2878189.

Mendes, P., Max, J., García-Silva, A., Bizer, C. (2011). DBpedia Spotlight: shedding light on the web of documents. In: *Proceedings of the 7th International Conference on Semantic Systems*, Graz, Austria, pp. 1–8. https://doi.org/10.1145/2063518.2063519.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Jeffrey, D. (2013a). Distributed representations of phrases and their compositionality. In: *Proceedings of the 27th Conference on Neural Information Processing Systems*, Lake Tahoe, USA, pp. 1–11. https://arxiv.org/abs/1310.4546.

Mikolov, T., Corrado, G.S., Chen, K., Dean, J. (2013b). Efficient estimation of word representations in vector space. In: *Proceedings of the International Conference on Learning Representations*, Scottsdale, USA, pp. 1–12. https://arxiv.org/abs/1301.3781v3.

Mohd, M., Jan, R., Shah, M. (2020). Text document summarization using word embeddings. *Expert Systems with Applications*, 143(1), 1–10. https://doi.org/10.1016/j.eswa.2019.112958.

Nooralahzadeh, F., Øvrelid, L., Lønning, J.T. (2018). Evaluation of domain-specific word embeddings using knowledge resources. In: *Proceedings of the 11th International Conference on Language Resources and Evaluation*. European Language Resources Association (ELRA), Miyazaki, Japan, pp. 1438–1445. https://www.aclweb.org/anthology/L18-1228.

Oliveira, J., Delgado, C., Assaife, A.C. (2017). A recommendation approach for consuming linked open data. *Expert Systems with Applications*, 72, 407–420. https://doi.org/10.1016/j.eswa.2016.10.037.

Park, S., Cho, J., Park, K., Shin, H. (2021). Customer sentiment analysis with more sensibility. *Engineering Applications of Artificial Intelligence*, 104, 104356. https://doi.org/10.1016/j.engappai.2021.104356.

Peeters, R., Bizer, C. (2021). Dual-objective fine-tuning of BERT for entity matching. *Proceedings of the VLDB Endowment*, 1410, 1913–1921.

Peeters, R., Primpeli, A., Wichtlhuber, B., Bizer, C. (2020). Using schema.org annotations for training and maintaining product matchers. In: *Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics*, Biarritz, France.

Pellissier, T.T., Weikum, G., Schanek, F. (2020). YAGO 4: a reasonable knowledge base. In: *Proceedings of the 17th International Conference on the Semantic Web (ESWC)*, Vol. 12123. Springer, Crete, Greece, pp. 583–596. https://doi.org/10.1007/978-3-030-49461-2_34.

Pennington, J., Socher, R., Manning, C.D. (2014). GloVe: global vectors for word representation. In: *Empirical Methods in Natural Language Processing*, pp. 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Petters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L. (2018). Deep contextualized word representations. In: *Proceedings of the Conference of the North American Chapter of the As-*

sociation for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics (ACL), New Orleans, USA, pp. 2227–2237. https://arxiv.org/abs/1802.05365v2.

Phan, M.C., Sun, A., Tay, Y., Han, J., Li, C. (2017). NeuPL: attention-based semantic matching andpair-linking forentity disambiguation. In: *Proceedings of the 2017 ACM Conference on Information Knowledge Management*, Singapore.

Pilehvar, M.T., Collier, N. (2016). Improve semantic representation for domain-specific entities. In: *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*. Association for Computational Linguistics (ACL), Berlin, Germany, pp. 12–16. https://aclanthology.org/W16-2902/.

Primpeli, A., Peeters, R., Bizer, C. (2019). The WDC training dataset and gold standard for large-scale product matching. In: *Proceedings of the 2019 World Wide Web Conference*, San Francisco, USA.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. (2019). Language models are unsupervised multitask learners. *Computer Science*, 1–24. https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

Rani, R., Lobiyal, D.K. (2022). Document vector embedding based extractive text summarization system for Hindi and English text. *Applied Intelligence*, 52, 9353–9372. https://doi.org/10.1007/s10489-021-02871-9.

Rehürek, R., Sojka, P. (2010). Software framework for topic modelling with large corpora. In: *Proceedings of the LREC Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pp. 45–50.

Reimers, N., Gurevych, I. (2019). Sentence-BERT: sentence embeddings using siamese BERT-networks. In: *Proceedings of Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics (ACL), Hong Kong, China, pp. 3982–3992. https://arxiv.org/abs/1908.10084.

Rezaeinia, S.M., Rahmani, R., Ghodsi, A., Veisi, H. (2019). Sentiment analysis based on improved pre-trained word embeddings. *Expert Systems with Applications*, 177(1), 139–147. https://doi.org/10.1016/j.eswa.2018.08.044.

Silva, R.M., Santos, R., Almeida, T., Pardo, T. (2020). Towards automatically filtering fake news in Portuguese. *Expert Systems with Applications*, 146, 1–14. https://doi.org/10.1016/j.eswa.2020.113199.

Smith, J., Plamada, M., Koehn, P., Callison, C., Lopez, A. (2013). Dirt cheap web-scale parallel text from the Common Crawl. In: *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria.

Songa, X., Mina, Y., Da-Xionga, L., Fengb, W.Z., Shua, C. (2019). Research on text error detection and repair method based on online learning community. *Procedia Computer Science*, 154, 13–19. https://doi.org/10.1016/j.procs.2019.06.004.

Sunitha, L., BalRaju, M., Sasikiran, J., Venkat Ramana, E. (2014). Automatic outlier identification in data mining using IQR in real-time data. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(6), 1–10.

Symseridou, E. (2018). The web as a corpus and for building corpora in the teaching of specialised translation. *FITISPOS International Journal*, 5(1), 60–82. https://doi.org/10.37536/FITISPos-IJ.2018.5.1.160.

Tukey, J.W. (1977). *Exploratory Data Analysis*. Reading Mass, New York.

Turian, J., Ratinov, L.A., Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics (ACL), Uppsala, Sweden, pp. 384–394. https://www.aclweb.org/anthology/P10-1040.

Valcarce, D., Landin, A., Parapar, J., Barreiro, A. (2019). Collaborative filtering embeddings for memory-based recommender systems. *Engineering Applications of Artificial Intelligence*, 85, 347–356. https://doi.org/10.1016/j.engappai.2019.06.020.

Vrandeĉić, D., Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10), 78–85. https://doi.org/10.1145/2629489.

Wang, B., Jay-Kuo, C.C. (2020). SBERT-WK: A Sentence Embedding Method by Dissecting BERT-based Word Models. https://arxiv.org/abs/2002.06652.

Xu, Y., Liu, X., Shen, Y., Liu, J., Gao, J. (2019). Multi-task learning with sample re-weighting for machine reading comprehension. In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics (ACL), Minneapolis, USA, pp. 2644–2655. https://arxiv.org/abs/1809.06963v3.

Yoo, S., Jeong, O. (2020). Automating the expansion of a knowledge graph. *Expert Systems with Applications*, 141(1), 1–10. https://doi.org/10.1016/j.eswa.2019.112965.

Zanzotto, F., Pennacchiotti, M. (2010). Expanding textual entailment corpora from Wikipedia using co-training. In: *Proceedings of the 2nd Workshop on Collaborative Constructed Semantic Resources*, Beijing, China.

Zhang, D., Yuan, Z., Liu, Y., Zhuang, F., H., C., Xiong, H. (2021). E-BERT: adapting BERT to e-commerce with adaptive hybrid masking and neighbor product reconstruction. https://arxiv.org/pdf/2009.02835.

Zhang, Z., Song, X. (2022). An exploratory study on utilising the web of linked data for product data mining. *SN Computer Science*, 4(15). https://doi.org/10.1007/s42979-022-01415-3.

Zhou, Y., Hu, X., Chung, V. (2022). Automatic construction of fine-grained paraphrase corpora system using language inference model. *Applied Intelligence*, 12(1). https://doi.org/10.3390/app12010499.

Zwicklbauer, S., Seifert, C., Granitzer, M. (2016). DoSeR: a knowledge-base-agnostic framework for entity disambiguation using semantic embeddings. In: *Proceedings of the 13th International Conference on Semantic Web Latest Advances and New Domains*, Heraklion, Crete.

Y. Blanco-Fernández obtained her PhD in telecommunications engineering from the University of Vigo in 2007 and currently serves as an associate professor at the same institution. Her research focuses on semantic reasoning in personalization systems, wireless ad hoc networks for mobile devices, machine learning, deep learning models, and natural language processing. She has authored 50+ JCR-indexed journal articles, 13 book chapters, and presented 93 communications at international conferences. She has also advised 5 doctoral theses and has contributed to 30+ competitively funded projects, both nationally and internationally (including H2020 and FP7). She has also engaged in 4 technology transfer contracts. Since 2021, she has held the position of deputy director at the Research Center for Telecommunication Technologies (atlanTTic).

A. Gil-Solla holds a degree in telecommunication engineering (1991) and earned his PhD in telecommunication (2000) from the University of Vigo. Currently, he holds the position of professor at the same institution, where he teaches in the Telecommunication programme. He has advised 5 PhD theses and supervised more than 20 undergraduate theses. He is a member of the Group of Services of the Information Society, which is part of the Department of Telematic Engineering at the University of Vigo. Throughout his career, he has been involved in over 40 national and international research projects, including FP7 and H2020 initiatives, with many of them being carried out in collaboration with industrial partners. His research interests focus on the design and development of intelligent systems for personalization of Internet and mobile applications. This includes automatic content recommendation, particularly utilizing Natural Language Processing techniques and other Machine Learning approaches involving neural networks. He has authored over 50 publications in journals indexed in the JCR, as well as more than 60 presentations at international conferences.

J.J. Pazos-Arias is a telecommunications engineer (1987) and holds a PhD in telecommunications engineering (1995) from the Polytechnic University of Madrid. He joined the University of Vigo in 1988 and has held the position of professor since 2009 in the Department of Telematics Engineering. Since June 2016, he has been a Numerary Academician of the Royal Academy of Sciences of Galicia. He co-authored 75 articles in JCR-indexed journals, contributed to 21 chapters in internationally recognized books, and presented over 150 communications at international congresses. Additionally, he has edited 2 books of research monographs and served as the principal investigator in 4 out of 5 projects of the National R&D Plan in which he participated. He has also advised 9 doctoral theses. He has been involved in numerous projects funded through competitive calls. Over the past decade, he has participated or is currently participating in two projects of the EU H2020 program, one project of the 7th EU Framework Program, one project under the Erasmus+ program, 4 projects funded through competitive national calls in collaboration with European partners, several regional projects, and 13 collaborative projects with companies funded through competitive national calls. He assumed the role of principal investigator in many of these projects. In terms of transferring research results, he has contributed to more than 30 technology transfer contracts and 17 contracts for training courses. Additionally, he has been the person responsible for overseeing many of these activities. In 1995, he founded the Information Society Services Group (GSSI) and continues to serve as its head. This group has attained the classification of a Reference Group within the R&D system of the Galician region, securing significant stable funding not tied to specific projects. He is also a member of the Research Center for Telecommunication Technologies (atlanTTic).

D. Quisi-Peralta received a degree in computer systems engineering from the Universidad Politécnica Salesiana (Ecuador) in 2013. Furthermore, he obtained a master's degree in advanced computer technologies from the University of Castilla-La Mancha (Spain) in 2015, as well as a master's in Strategic Management of Communication Technologies from the University of Cuenca (Ecuador) in 2017. Currently, he is a PhD student at the School of Telecommunications Engineering at the University of Vigo (Spain). He works as the director of the ICT department at Livingnet and serves as an external researcher for PUCE (Pontificia Universidad Católica del Ecuador) and UPS (Universidad Politécnica Salesiana). His research interests encompass the application of AI technologies, data mining, ontologies, large language models, computer vision, and application development.