UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA
DEPARTMENT OF INFORMATICS, SYSTEMS AND COMMUNICATION
PH.D. IN COMPUTER SCIENCE

Integrating Word Embeddings and Taxonomy Learning for Enhanced Lexical Domain Modelling

Supervisor: Prof. Fabio Mercorio
Co-Supervisor: Prof. Mario Mezzanzanica
Tutor: Prof. Simone Bianco

Candidate:
ANNA GIABELLI
NUM. 791989

Academic Year 2022-2023

Abstract

This PhD thesis aims to integrate lexical taxonomies and word embeddings to develop novel methodologies for enhancing Natural Language Processing representations. Lexical taxonomies serve as a natural means of organising human knowledge in a hierarchical way and offering formal descriptions of concepts and their relationships, supporting both syntactic and semantic exchanges. On the other hand, word embeddings are vector representations of words that capture linguistic patterns and lexical semantics from extensive corpora based on the idea that words with similar contexts tend to have similar meanings.

This research explores the conjunction of word embeddings with the structure of lexical taxonomies, enabling the choice of word embeddings that fit the hierarchical structure of the concepts they represent. Additionally, word embeddings can aid in updating taxonomies to accommodate evolving languages and knowledge domains. They facilitate the incorporation of new concepts in the appropriate taxonomic positions by leveraging vast textual data. Moreover, word embeddings can be valuable for aligning and linking taxonomies, which is crucial when multiple taxonomies within a single domain, built by different institutions for varying purposes, need to communicate effectively.

The thesis is divided into several parts.

Part I introduces the two fundamental subjects of word embeddings and lexical taxonomies.

Part II focuses on two methods for evaluating word embeddings. One method, `TaxoVec`, is a framework to select taxonomy-aware word embeddings leveraging a measure of taxonomic semantic similarity (the `HSS`), while `vec2best` offers a general evaluation framework for word embeddings without a specific taxonomy. It provides a comprehensive evaluation metric called the *PCE* (*Principal Component Evaluation*) for each model.

Part III details two methodologies for enhancing and aligning lexical taxonomies using word embeddings. `NEE` enables taxonomy enrichment by estimating data conformity to a given taxonomy and identifying new entities and concepts. `WETA` is a domain-independent method for automatic taxonomy alignment, combining hierarchical similarity and classification tasks into a scoring function.

Part IV showcases the practical applications of the proposed methodologies in the context of Labour Market Intelligence.

This research contributes to Natural Language Processing by providing innovative techniques for enhancing language representation and knowledge, ultimately benefiting various applications in this domain.

Acknowledgments

First and foremost, I want to thank Prof. Mario Mezzanzanica and Prof. Fabio Mercorio, my PhD supervisors, because without them I would not have chosen this path, and I am grateful for that. I also want to thank Prof. Simone Bianco for the technical support he provided me during these years.

I would like to also thank all my colleagues in CRISP for their help and backup during these years.

Lastly, I want to thank my family for their support throughout my life.

Contents

Part I

Introduction and Background

1

Integrating Word Embeddings and Taxonomies

In recent years, lexical taxonomies and distributional semantics have gained momentum in Natural Language Processing (NLP) applications.

Lexical taxonomies are a natural way to organise human knowledge in a hierarchical form and to provide a formal description of concepts and their relations, supporting syntactic and semantic exchanges. Contextually, word embeddings have gained remarkable popularity in computational linguistics. They are vector representations of words based on the hypothesis that words occurring in a similar context are prone to have a similar meaning.

Despite their wide usage, finding a unified framework accounting for knowledge-based and distributional resources is still an open problem.

The idea is that word embeddings, being able to extract linguistic patterns and lexical semantics from large corpora could benefit from assessing if the structure of the embeddings fits the structure of lexical taxonomies. That could allow researchers to choose a word embedding trained on a large corpus of texts to represent words in a vector space that also fits the hierarchical structure of the concepts it embeds. That is particularly relevant given

the high sensitivity of word embeddings to small changes in hyper-parameters during their generation [37] and given that the selection of an embedding over another affects the quality of the overall process or system in which they are used.

On the other hand, taxonomies are not fixed but need updating due to the constant changes in languages and every field of knowledge. The most adopted approaches to enrich or extend standard taxonomies lean on expert panels and surveys; those are used to identify and validate which terms should be added to a taxonomy. The approach relies on human knowledge, and this makes the process error-prone - different experts can have different opinions - and manually identifying new concepts and where to put those in the taxonomy structure is time-consuming and costly. Resorting to word embeddings can help reduce these problems, allowing to leverage big text data available to try and enrich existing taxonomies with new concepts in the right places. A similar approach could also be valuable in aligning and linking taxonomies since usually, within a single domain, multiple taxonomies are built by different institutions for different purposes and need to be talkative to each other to allow interactions from one to the other to integrate diverse data. The manual mapping of two taxonomies by domain experts is a time-consuming and costly task, often leading to inaccuracies. Word embeddings can be used also in this context to allow automatic aligning taxonomies based on the semantic meaning of the concepts that can be learnt from large corpora.

1.1 Contribution

In this thesis, the symbiotic utilisation of both lexical taxonomies and word embeddings is explored, fostering a mutually beneficial relationship between them. More specifically, in the work described in this thesis:

- We propose a novel method to select taxonomy-aware word embeddings [68]: `TaxoVec` is a framework that leverages taxonomic semantic similarity - thanks to the `HSS`, a measure of semantic similarity between taxonomic words - to evaluate word embedding models.
- We develop a novel method for taxonomy enrichment through the use of word embeddings [69, 70]: `NEE` leverages word embeddings to estimate the degree to which data

conforms to a given taxonomy, identifying new entities and concepts for the taxonomy itself.

- We present a novel method for taxonomy alignment through the use of word embeddings [67]: `WETA` is a domain-independent, knowledge-poor method for automatic taxonomy alignment employing a scoring function, which merges the score of a hierarchical method based on cosine similarity and the score of a classification task.
- We propose a general evaluation method for word embeddings when a taxonomy is missing in the field: `vec2best` furnishes the user with an extensive evaluation of word embedding models. It represents a framework for evaluating word embeddings trained using various methods and hyper-parameters on a range of tasks from the literature. The tool yields a holistic evaluation metric for each model called the *PCE* (*Principal Component Evaluation*). It is crucial to have a reliable measure of evaluation that can produce a performant semantic representation based on the intended scope and the information they have to embed because of the high sensitivity of word embeddings to small changes in hyper-parameters during their generation.

In this work, we did not consider contextual word embedding because, as shown by Asudani et al. [12], static word embeddings are still relevant and frequently used in Natural Language Processing tasks (see the last Section of Chapter 2). Moreover, contextual embeddings are more computationally expensive to train than static embeddings [127] and, when considering a specific application, as is frequently the case, the different meanings of words are restricted by default, making the trade-off between contextual and static embeddings favours the latter.

1.2 Working Example

To give a better idea of the relevance of this paper to the current state-of-the-art, let us consider the 2012 ACM Computing Classification System (CCS)[1], a taxonomy devised by the Association for Computing Machinery (ACM) that serves as the de facto standard classification system for the computing field.

[1] https://dl.acm.org/ccs

The CCS is used in the ACM Digital Library (DL)[2] to index content for subject-oriented searching, to find similar documents, to create author expertise profiles, to identify robust research areas in Institutional Profiles, and to create the topical tag clouds found in aggregated ACM Special Interest Groups and conference views. Outside the DL, researchers and institutions use the CCS in their own applications and research projects [126].

The system mentioned above went through seven revisions, the first version being published in 1964, and revised versions appearing in 1982, 1983, 1987, 1991, 1998, and the current version in 2012. such a revision reflects the need for keeping a taxonomy that provides a map of the field of computing in all its breadth up-to-date. For the 2012 version of the CCS, first drafts were created using inter alia user search logs from the ACM Digital Library, machine analysis of DL texts and author-supplied keyword occurrences, and manual examination of extant computer science taxonomies. ACM's domain experts used these drafts as their starting point, and the final taxonomy reflects a year-long team effort that included two review stages and many iterations [126].

The computing field is dynamic and constantly changing therefore it is important to optimise the process of updating the standard taxonomy. Word embeddings could be used to propose an update of the CCS, using titles and abstracts of papers to train the models and using the similarity of novel topics to suggest to some domain experts where to add them to the existing taxonomy.

On the other hand, the CCS can be used for selecting the word embedding model that best encodes the structure of the CCS. For example, if we are working with titles and abstracts of papers, word embeddings could be used for various NLP tasks, e.g. as a vector representation that will be fed to the neural network for the automation of the citation screening process [144], or to perform topic modelling [136]. In this context, CCS could be used to select the word embedding that best represents the Computer Science topics and the hierarchical relations between them, allowing for better results over these tasks.

[2] https://dl.acm.org

1.3 Thesis Structure

The thesis is composed of four main parts, organised as follows.

Part I It is intended to introduce the two main topics on which this thesis is based: word embeddings and lexical taxonomies. In Chapter 1, we introduce the symbiotic utilisation of lexical taxonomies and word embeddings, fostering a mutually beneficial relationship between them. In Chapter 2, we define what are word embeddings and describe some of the major methodologies used to create them. Lastly, in Chapter 3, we introduce the concept of taxonomies, giving a formal definition that will be useful in the following chapters and defining state-of-the-art taxonomy-based semantic similarity measures.

Part II It focuses on two methods for the intrinsic evaluation of word embeddings. In Chapter 4, we introduce the topic of word embedding evaluation and present some relevant works from other authors. In Chapter 5, we propose the `HSS` - a measure of semantic similarity between concepts - and `TaxoVec` - a framework to select taxonomy-aware word embeddings. Lastly, in Chapter 6, we propose the `vec2best` tool, a unified approach to several state-of-the-art intrinsic evaluation tasks over different benchmarks, which produces a comprehensive measure of evaluation for each model called the *PCE (Principal Component Evaluation)*.

Part III It describes two methodologies for enhancing and aligning lexical taxonomies thanks to word embeddings. In Chapter 7, we introduce the topic of taxonomy induction, taxonomy enrichment, and taxonomy alignment. In Chapter 8, we present `NEE`, a framework designed to identify novel entities in a certain domain and to put them in the right place within the taxonomy. Lastly, in Chapter 9, we propose `WETA`, a domain-independent and knowledge-poor method for automatic taxonomy alignment via word embeddings.

Part IV The last part shows some applications of the methodologies proposed in the two previous parts in a specific context: Labour Market Intelligence. In Chapter 10, we introduce the field of study of Labour Market Intelligence and the projects in which the works in the later chapters are framed. In Chapter 11, we propose two tools, `NEO` and `NES`, used to automatically enrich the European Skills, Competences, Qualifications, and Occupations taxonomy (ESCO) with new terms from a free text corpus. In Chapter 12, we propose `JoTA` (Job Taxonomy Alignment), a framework used to align the Italian taxonomy of occupations, CP and the European ESCO occupation pillar taxonomy. Lastly, in Chapter 13 we present the conclusion of this work.

2

Word Embeddings

The task of learning a representation for words and documents from a corpus of texts is a crucial part of Natural Language Processing (NLP) tasks. Over the years, researchers proposed multiple ways of representing words using vectors, which have an intuitive interpretation, can be the subject of functional operations, and lend themselves well to be used in many Machine Learning (ML) algorithms.

The earlier techniques are called conventional models, also known as count-based or frequency-based models. Later, the distributional representation models, also called static word embeddings, gained relevance: the context of a word is used to determine its meaning in a sentence and assign a single vector to it. More recently, contextual models create word representations that are not unique but multiple and that are directly computed from the context.

2.1 Conventional Word Representation Models

2.1.1 Bag of Words (BoW)

In the Bag of Words model, the text is an unordered collection of words, with no attention to grammar or even to the word's order.

For example, let us consider the two statements:

Statement 1: One cat is sleeping, and the other one is running.

Statement 2: One dog is sleeping, and the other one is eating.

The representation of those is:

	one	cat	is	sleeping	and	the	other	dog	running	eating
S1	2	1	2	1	1	1	1	0	1	0
S2	2	0	2	1	1	1	1	1	0	1

BoW suffers some limitations, such as sparsity: if the length of a sentence is large, it takes significant time to obtain its vector representation and to get sentence similarity. Another limitation is that frequent words have more power: their frequency count increases, increasing their similarity scores. Lastly, ignoring word orders leads to losing the sentence's contextual meaning [12].

2.1.2 N-grams

The n-gram model is similar to the BoW, but instead of words, we consider a contiguous sequence of n tokens. For $n = 1, 2, 3, \ldots$, it is termed as 1-gram, 2-gram, and 3-gram, also termed as uni-gram model, bi-gram, and tri-gram.

Considering again the two sentences from the BoW example, their bi-gram level representation is shown in the example below.

This model still presents the limitations of the BoW.

	one cat	cat is	is sleeping	sleeping and	and the	the other
S1	1	1	1	1	1	1
S2	0	0	1	1	1	1
	other one	one is	is running	one dog	dog is	is eating
S1	1	1	1	0	0	0
S2	1	1	0	1	1	1

2.1.3 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is used to assess the importance of a term in relation to a collection of documents. In contrast with term frequency (TF) - which measures how frequently a term occurs in a document - it also allows finding terms that are frequent in the considered document and infrequent in other documents of the collection (through the IDF).

If we consider the two sentences from the BoW example, and consider those as two different documents, we can compute the TF-IDF for each word in each document:

		one	cat	is	sleeping	and	the	other	running
S1	TF	0.2	0.1	0.2	0.1	0.1	0.1	0.1	0.1
	IDF	0	0.3	0	0	0	0	0	0.3
	TF-IDF	0	0.03	0	0	0	0	0	0.03
		one	dog	is	sleeping	and	the	other	eating
S2	TF	0.2	0.1	0.2	0.1	0.1	0.1	0.1	0.1
	IDF	0	0.3	0	0	0	0	0	0.3
	TF-IDF	0	0.03	0	0	0	0	0	0.03

The TF-IDF is computed as:

$$tf - idf(t,d,D) = tf(t,d) \times idf(t,D) \tag{2.1}$$

with $tf(t,d)$ relative frequency of term t within document d:

$$tf(t,d) = \frac{f_{td}}{\sum_{t' \in d} f_{t'd}} \tag{2.2}$$

and $idf(t, D)$ logarithm of the fraction of the total number of documents (N) and the number of documents containing the term.

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2.3}$$

2.2 Word Embedding Models

Word embeddings are vector representations of words based on the hypothesis that words occurring in a similar context are prone to have a similar meaning.

Word embeddings can be defined as lookup tables mapping words to vectors of real numbers [46]. They are functions giving continuous vector representations in $\mathbb{R}^D$ for elements of a set V (e.g. a set of words or tokens). Formally, a word embedding e can be represented as:

$$\begin{aligned} e: \quad & V \to \mathbb{R}^D \\ & w \mapsto e(w) = v_w^e \end{aligned} \tag{2.4}$$

To estimate the word embedding function e, we need to perform an optimisation process on a large sample of language data on an arbitrary task. In the end, the embedding has accumulated information from the corpus for each word vector, leading to noticeable geometric relationships [139].

A common distinction between different types of embedding is between static and contextualised embeddings:

- Static word embeddings use the distributional hypothesis to learn global and constant vectors. They represent a word by a unique vector condensing every local usage of the word in the training corpus. Two powerful methods to induce static word embeddings are neural networks training [45, 107] and co-occurrence matrix factorisation [117, 95]. These word embeddings can be further divided into:
  - Euclidean word embeddings use Euclidean geometry as mathematical support to embed vectors. Yet, due to the intrinsic properties of this geometry, it may be challenging to embed asymmetric information. These types of embeddings

preferably incorporate symmetric knowledge such as semantic synonymy or analogy [139].

  - Non-Euclidean geometries provide a solution to embed asymmetric relations such as entailment or logical thinking, but are less used compared to the previous ones.

- Contextualised word embeddings are models providing variable vectors. Word representations are multiple and are directly computed from their context. The context of a word is usually composed of the words surrounding it. For contextualised embeddings, non-Euclidean embeddings have not yet been investigated.

The most prominent word embedding models discussed in the following sections are summarised in Table 2.1.

Table 2.1 The most prominent word embedding models published from 2013 to 2020 [12].

Embedding approach	Year	Organisation	References
word2vec	2013	Google Inc	[106, 107]
GloVe	2014	Stanford University	[117]
fastText	2016	Facebook AI Research Lab	[85]
ELMo	2017	Allen Institute of AI	[118]
GPT	2018	OpenAI	[120]
BERT	2018	Google AI Lab	[50]
GPT2	2018	OpenAI	[121]
GPT3	2020	OpenAI	[31]

2.3 Euclidean Static Word Embeddings

2.3.1 Word2vec

The word2vec algorithm [107] uses a two-layer feed-forward neural network architecture, and in the original article, two models were presented, which are Continuous Bag of Word (CBOW) and skip-gram (SG). The difference between them is the task on which they are trained: CBOW aims to predict a target word given its context, while skip-gram aims to forecast, given a target word, its context.

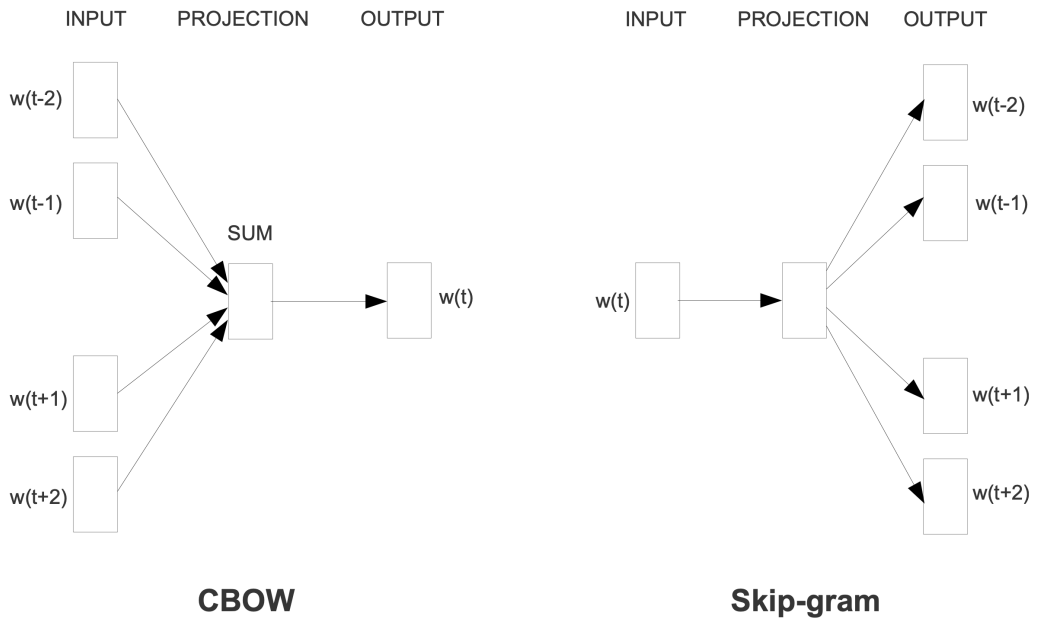


Figure 2.1 The CBOW and skip-gram model architectures.

Focusing on skip-gram, the model uses each current word as an input to a log-linear classifier with a continuous projection layer and predicts words within a specific range before and after the word. More formally, given a sequence of training words $w_1, w_2, w_3, \ldots, w_T$, the objective of the skip-gram model is to maximise the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \tag{2.5}$$

where c is the size of the training context (which can be a function of the centre word w_t).

Some important parameters and settings for the training of word2vec are:

- The training can be executed with the hierarchical softmax, which is a computationally efficient approximation of the full softmax: instead of evaluating W output nodes in the neural network to obtain the probability distribution, it is needed to evaluate only about $log_2(W)$ nodes [107].
- The training can also be executed with the negative sampling method, which approaches the maximisation problem by minimising the log-likelihood of sampled negative instances [107].

- Words with a frequency above or below a certain threshold may be sub-sampled or removed to speed up training.
- The dimensionality of the vectors affects the results: the quality of the word embeddings increases with the increase of the dimensionality, but after reaching some point, the marginal gain diminishes.
- The size of the context window determines how many words before and after a given word are included as context words of the given word.

2.3.2 FastText

Bojanowski et al. [24] developed a version of the continuous skip-gram model called fastText. One of its major improvements to word2vec is to consider sub-word information by representing each word as the sum of its character *n*-gram vectors. Formally, given a word w, and a dictionary of size G, G_{w} is the set of *n*-grams of size G appearing in w. Denoting as z_g the vector representation of the *n*-gram g, w will be represented as the sum of the vector representation of its *n*-grams and the score associated to the word w as:

$$\int(\mathrm{w},\mathrm{c}) = \sum_{g \in G_{\mathrm{w}}} z_g^{\top} v_{\mathrm{c}} \tag{2.6}$$

where v_{c} is the vector representing the context. This simple representation allows one to share information between words, and this makes it useful to represent rare words, typos, and words with the same root. Moreover, it allows the computation of representations for words not seen during the training phase, called Out Of Vocabulary (OOV) words.

2.3.3 GloVe

GloVe [117] is a global log-bilinear regression model for the unsupervised learning of word representations. The model leverages statistical information by training only on the nonzero elements in a word co-occurrence matrix rather than the entire sparse matrix or the individual context windows in a large corpus.

The idea is to understand the relationship between words by studying the ratio of their co-occurrence probabilities with various probe words. Compared to the raw probabilities,

the ratio is better at distinguishing relevant from irrelevant terms. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. Since the logarithm of a ratio equals the difference of logarithms, this objective associates the logarithm of ratios of co-occurrence probabilities with vector differences in the word vector space.

2.4 Non-Euclidean Static Word Embeddings

Another category of word embedding models is that of hyperbolic embeddings, which learn embeddings in hyperbolic vector spaces that, due to their geometry (hyperbolic space can be thought of as a continuous version of trees), are more equipped to model hierarchical structures. In [110], the authors introduced an approach for learning hierarchical representations of symbolic data by embedding them into hyperbolic space, more precisely into an n-dimensional Poincaré ball, such that their distance in the embedding space reflects their semantic similarity. Other non-Euclidean embeddings have been explored (see e.g. [111, 63]).

2.5 Contextualised Word Embeddings

2.5.1 Embeddings from Language Models

The first contextualised embedding model was ELMo [118], which, after pre-training on a large dataset, transfers the internal representations from the bidirectional language models (BiLSTM) to a downstream model of interest as contextual word representations.

ELMo considers the complete sentence when assigning an embedding to each word: it employs a bidirectional design, embedding depending on the sentence's next and preceding words. Given a sequence of N tokens $(t_1, t_2, \ldots, t_N)$, the aim is to find the language model's highest probability in both directions. The likelihood of the sequence is computed using a forward language model, which models the chance of token t_k considering the history $(t_1, t_2, \ldots, t_k)$. A backward language model is identical to a forward language model but goes backwards through the sequence, anticipating the previous token based on the future context [12].

2.5.2 Generative Pre-Training

More recent models use transformer decoders, such as GPT, instead of LSTM or recurrent networks. Even when the number of parameters is increased, transformer-based models tend to be more effective [139].

The GPT model uses a fine-tuning technique and only uses task-specific parameters that have been trained on downstream tasks. Moreover, GPT uses a one-way language model, to extract features, whereas ELMo employs a BiLSTM.

A standard language modelling objective for a sequence of tokens $(t_1, t_2, \ldots, t_N)$ to maximise the likelihood is:

$$L_1(X) = \sum \log P(t_i | t_{i-N}, \ldots, t_{i-1}; \theta) \tag{2.7}$$

The language model employs a multi-layer transformer decoder with a self-attention mechanism to anticipate the current word through the previous N words. To achieve an adequate distribution over target words, the GPT model employs a multi-headed self-attention operation over the input contextual tokens, accompanied by position-wise feed-forward layers [12].

2.5.3 Bidirectional Encoder Representations from Transformers

BERT [50] stands for Bidirectional Encoder Representations from Transformers. BERT is designed to pre-train deep bidirectional representations from an unlabelled text by conditioning on the left and right contexts in all layers, making it fine-tunable with an additional output layer.

BERT employs masked language modelling to optimise and combine position embedding with static word embeddings as model inputs. It follows frameworks for both pre-training and fine-tuning.

Initially, the model is trained on unsupervised learning from two pre-training tasks: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). MLM uses a MASK token hiding a target word in a sentence, and the objective is to reconstruct the original token given the whole context. NSP is a classification problem where the model has

to tell whether a sentence can follow another. After the training phase, the representation for the words is taken from the output of the encoder [139].

The model is then fine-tuned by initialising it using the pre-trained parameters and then fine-tuning all parameters using labelled data from the downstream jobs.

BERT uses a deep, pre-trained neural network with transformer architecture to create dense vector representations for natural language.

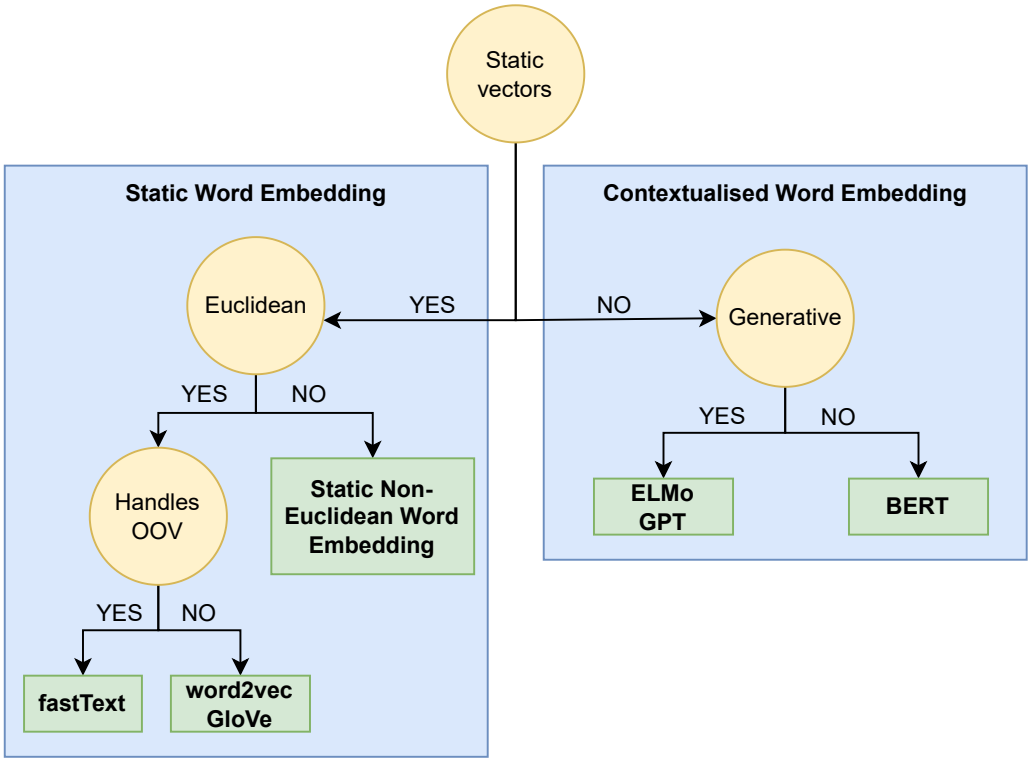


Figure 2.2 Relations between word embeddings based on some basic properties (adapted from [139]).

2.6 Comparative Analysis of Word Embedding Models for Text Analytics Tasks

The different properties of the word embedding techniques described before are summarised in Figure 2.2.

The performance of these word embedding techniques for various text analytics tasks was observed in the review from Asudani et al. [12], and the results are shown in Fig. 2.3.

The study shows that the domain-specific word embedding performance is higher than the generalised embedding approach for performing tasks related to text analytics. For example, for the text classification task, the CBOW model of word2vec and domain-specific embedding performance are similar; GloVe, fastText, and BERT embedding models show considerable performance and are limited to a few applications. The researchers utilise the ELMo and GPT models for text classification tasks in minimal circumstances.

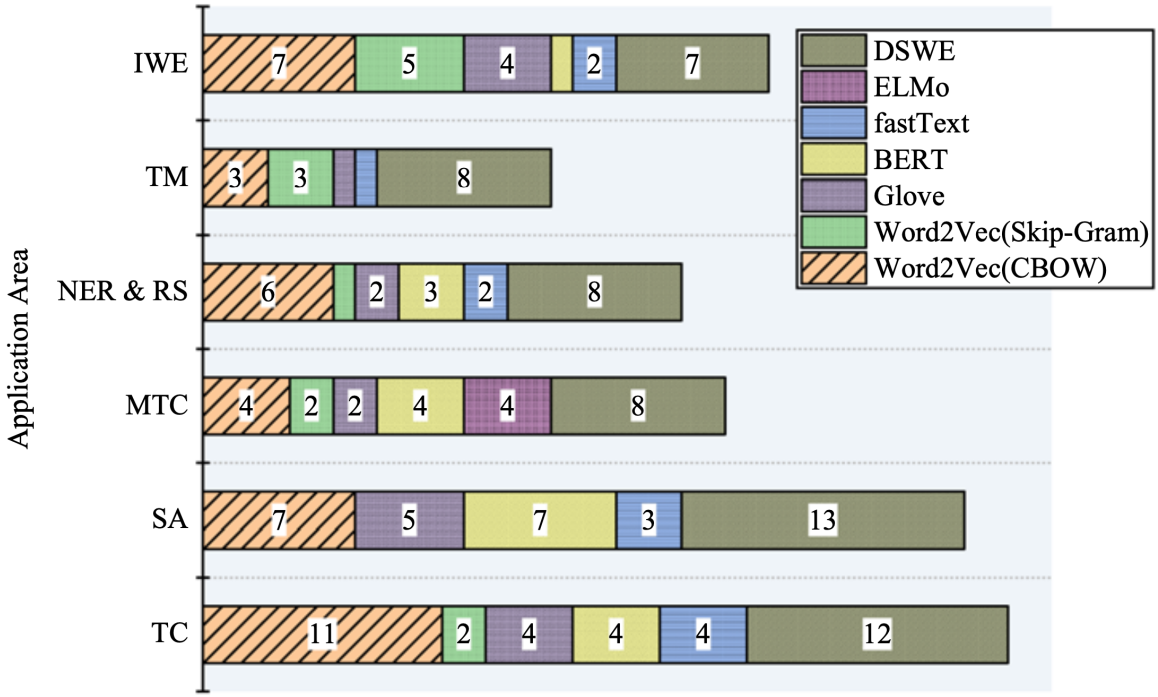


Figure 2.3 Performance of word embedding models based on the review from Asudani et al. [12]. TC: text classification, SA: sentiment analysis, MTC: medical text classification, NER & RS: named entity recognition and recommendation system, TM: topic modelling, IWE: impact of word embedding, DSWE: domain-specific word embedding.

It emerges from the review, that domain-specific word embedding achieves the first preference as the most suitable embedding for most of the application areas related to text analytics. The CBOW model also achieves the first preference for performing text classification tasks, whereas GloVe, fastText, and BERT models achieve the second preference, as shown in Tab. 2.2. The CBOW and BERT model achieves the second preference for performing the sentiment analysis task. The CBOW, BERT, and ELMo models achieve second preference for performing biomedical text mining tasks. The CBOW model is the second choice for performing operations on the NER and recommendation system. The Skip-Gram

and GloVe model achieves the second preference to perform topic modelling-related tasks. The domain-specific word embedding and CBOW embedding models are recommended as the first preferences, whereas the Skip-Gram model is recommended as a second preference to analyse the impact of the word embedding model on text analytics tasks [12].

Table 2.2 The most suitable word embedding models for some relevant text analytics tasks [12]. TC: text classification, SA: sentiment analysis, MTC: medical text classification, NER & RS: named entity recognition and recommendation system, TM: topic modelling, IWE: impact of word embedding, DSWE: domain-specific word embedding.

Application	**word2vec**						
	CBOW	**SG**	**GloVe**	**BERT**	**fastText**	**ELMo**	**DSWE**
TC	☑		✓	✓	✓		☑
SA	✓			✓			☑
MTC	✓			✓		✓	☑
NER & RS	✓						☑
TM		✓	✓				☑
IWE	☑	✓					☑

Notes ☑: first preference, ✓: second preference.

This review shows how static word embeddings are still relevant and frequently used in Natural Language Processing tasks, and this is the main reason why we did not consider contextual word embedding. Another reason is that contextual embeddings are more computationally expensive to train than static embeddings, requiring a massive amount of data and computational power [127], making the evaluation of those embeddings potentially as expensive as their generation. Moreover, when considering a specific application, as is frequently the case, the different meanings of words are restricted by default, e.g. with financial data, there is no context referring to the river's bank, and the word bank has just one meaning in each context of those data. For this reason, in many applications, the trade-off between contextual and static embeddings favours the latter, considering the amount of data and computational power required by the former.

3

Lexical Taxonomies

3.1 Background on Taxonomies

A taxonomy is a semantic hierarchy that organises concepts by *is-a* relations, which express the notion that an entity is an example of a type (for example, "John is a bachelor") or that a type is a sub-type of another type (for example, "A dog is a mammal") [30].

In linguistics, *is-a* relations are called hyponymy: when a term describes some subset of a category defined by another term, the bigger term is called a *hypernym* with respect to the smaller, and the smaller is called a *hyponym* with respect to the larger. In other words, a concept represented by a lexical item L_0 is said to be a hyponym of the concept represented by a lexical item L_1 if native speakers of English accept sentences constructed from the frame. An L_0 is a (kind of) L_1. Here L_1 is the hypernym of L_0, and the relationship is reflexive and transitive, but not symmetric [77].

Taxonomies exhibit the capability of improving many NLP and IR tasks [149], such as query understanding [79], personalised product recommendation [80, 160], question answering [158]. It also supports a variety of real-world applications, including information management [112, 72], biomedical systems [91, 41], and e-commerce [1]. More specifically,

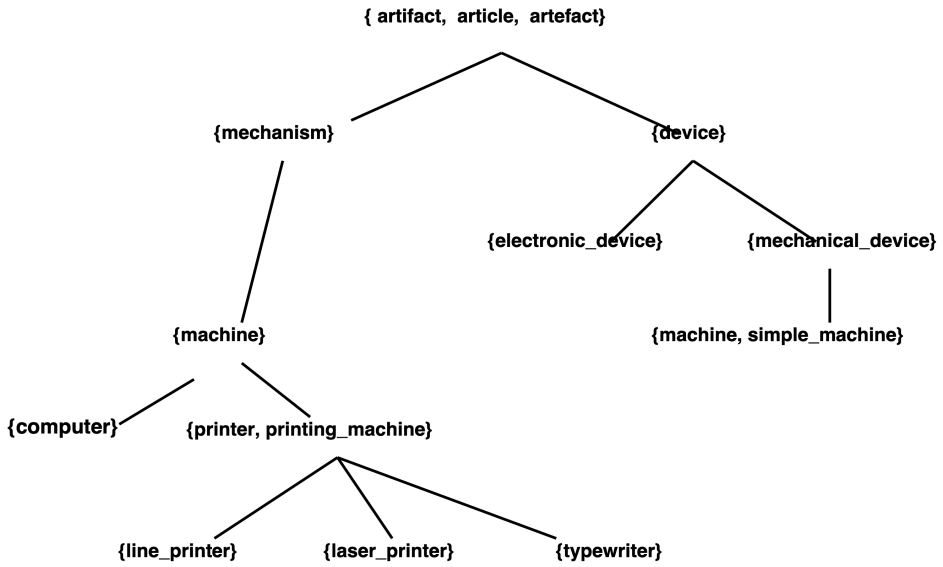


Figure 3.1 A Fragment of the WordNet Noun Hierarchy [77].

many online retailers (e.g., eBay and Amazon) organise products into categories of different granularities so that customers can easily search and navigate this category taxonomy to find the items they want to purchase [134].

Human knowledge is inherently organised in the form of semantic, content-specific hierarchies. Large-scale taxonomies such as Wikipedia Categories[1], Freebase [25] and WordNet [57] are crucial sources of structured knowledge useful for various natural language processing applications. However, although these hierarchies are well developed, they are primarily generic and laborious to augment and maintain, with new concepts and relations from newly emerging or rapidly evolving domains such as public health and current affairs [145].

As an example, Figure 3.1 indicates the portion of the hyponymy relation in WordNet's noun hierarchy that has to do with printers and devices.

3.1.1 A Formal Definition of Taxonomy

In this section, we introduce a formal definition of taxonomy, relying on the formalisation proposed by [102].

Definition 3.1.1 (Taxonomy) *A taxonomy $\mathcal{T}$ is a 4-tuple $\mathcal{T} = (\mathcal{C}, \mathcal{W}, \mathcal{H}^c, \mathcal{F})$.*

[1] https://en.wikipedia.org/wiki/Portal:Contents/Categories

- *$\mathcal{C}$ is a set of concepts $c \in \mathcal{C}$ (aka, nodes) that can be classified in p different hierarchical levels: $\mathcal{C}_1, \ldots, \mathcal{C}_p$;*
- *$\mathcal{W}$ is a set of words (or entities, or leaf concepts) belonging to the domain of interest; each word $w \in \mathcal{W}$ can be assigned to none, one or multiple concepts $c \in \mathcal{C}$.*
- *$\mathcal{H}^c$ is a directed taxonomic binary relation existing between concepts, that is $\mathcal{H}^c \subseteq \{(c_i, c_j) \mid (c_i, c_j) \in \mathcal{C}^2) \wedge i \neq j\}$. $\mathcal{H}^c(c_1, c_2)$ means that c_1 is a sub-concept, or hyponym, of c_2, while c_2 is the hypernym of c_1, meaning c_2 has a broader meaning and constitutes a category into which c_1 falls. The relation $\mathcal{H}^c(c_1, c_2)$ is also known as is-a relation (i.e., c_1 is-a sub-concept of c_2).*
- *$\mathcal{F}$ is a directed binary relation mapping words into concepts, i.e. $\mathcal{F} \subseteq \{(c, w) \mid c \in \mathcal{C} \wedge w \in \mathcal{W}\}$. $\mathcal{F}(c, w)$ means that the word w is an entity of the concept c.*

$\mathcal{T}$ might be represented as a Directed Acyclic Graph (DAG). Therefore, the concepts at the most specific level have an in-degree of 0*, i.e. they don't have any incoming edge. We refer to those concepts as leaf concepts, which are concepts representing different entities or words. Note that - in several taxonomies - the terms representing leaf concepts are also item words, while concepts at a higher level are not.*

3.2 Taxonomy-based Semantic Similarity

In this section, we cover the topic of how to measure semantic similarity between words in a taxonomy, expressed as a similarity between the concepts to which the two words belong. That is an important task concerning the taxonomies and will be useful in the definition of a new method for measuring semantic similarity in taxonomies in Chapter 5.

The measures of semantic similarity can be roughly divided into two main categories: those that exploit the path connecting two concepts and those that are based on the Information Content (IC) of the concepts. The most significant measures belonging to these two categories, as assessed by different researchers [93, 9], are the following.

Path-based Measures These measures employ the path connecting two concepts to estimate their similarity.

The simplest approach is to use the *shortest path* to assign a similarity score:

$$sim_{sp}(c_1, c_2) = \frac{1}{\phi(c_1, c_2) + 1} \tag{3.1}$$

where $\phi(c_1, c_2)$ is the shortest path between c_1 and c_2.

Leacock and Chodorow [94] scale the path similarity by the depth of the taxonomy:

$$sim_{lc}(c_1, c_2) = -\log \frac{\phi(c_1, c_2)}{2 \times max\_depth} \tag{3.2}$$

Wu and Palmer [153] consider the position of $LCA(c_1, c_2)$, the Lowest Common Ancestor of c_1 and c_2:

$$sim_{wup}(c_1, c_2) = \frac{2 \times \phi(r, LCA)}{\phi(c_1, LCA) + \phi(c_2, LCA) + 2 \times \phi(r, LCA)} \tag{3.3}$$

Where r is the root node. Note that, to simplify the notation, we refer to $LCA(c_1, c_2)$ simply as LCA. For this class of methods in the case of polysemy, i.e. words belonging to several concepts, the minimum $\phi(c_1, c_2)$ is considered, thus the maximum similarity.

Information Content-based Measures The IC-based approach was introduced by Resnik [124]. According to information theory, the IC (or self-information) of a concept $c \in \mathcal{C}$ can be approximated by its negative log-likelihood:

$$IC(c) = -\log p(c) \tag{3.4}$$

Where $p(c)$ is the probability of encountering the concept c.

In the case of a taxonomy, $p(c)$ is monotonic and increases with the rank of the taxonomy: if $c_1\ is-a\ c_2$ then $p(c_1) \leq p(c_2)$. Moreover, the probability of the root node is 1. Concepts probabilities are computed simply as relative frequencies in a text corpus:

$$\hat{p}(c) = \frac{freq(c)}{N} = \frac{\sum_{n \in words(c)} count(n)}{N} \tag{3.5}$$

Where $words(c)$ is the set of words subsumed by concept c, and N is the total number of occurrences of words in the corpus that are also present in the taxonomy. Therefore *Resnik* defines the similarity between two concepts c_1 and c_2 as:

$$sim_{res}(c_1, c_2) = IC(LCA) \tag{3.6}$$

The similarity between the two words is:

$$sim_{res}(w_1, w_2) = \max_{\substack{c_1 \in s(w_1), \\ c_2 \in s(w_2)}} IC(LCA) \tag{3.7}$$

Where $s(w_1)$ and $s(w_2)$ are all the possible senses of w_1 and w_2 respectively.

Jiang-Conrath [83] built a measure of similarity using the self-information of c_1 and c_2 as well:

$$sim_{jcn}(c_1, c_2) = \frac{1}{IC_{res}(c_1) + IC_{res}(c_2) - 2 \times IC_{res}(LCA)} \tag{3.8}$$

And a similar measure is built by Lin [97]:

$$sim_{lin}(c_1, c_2) = \frac{2 \times IC_{res}(LCA)}{IC_{res}(c_1) + IC_{res}(c_2)} \tag{3.9}$$

Similarly to Resnik, these last two methods consider the two concepts with the highest Resnik similarity in the case of multiple word senses.

Other IC-based measures, sometimes called *intrinsic IC-measures*, use the structure of the taxonomy, instead of an external corpus frequency, to compute $\hat{p}(c)$. For instance Seco et al. [132] compute the IC(c) as:

$$IC_{seco}(c) = 1 - \frac{\log(|descendants(c)| + 1)}{\log N_c} \tag{3.10}$$

where N_c is the number of concepts in the taxonomy and $|descendants(c)|$ the number of sub-concepts of c.

They have two main drawbacks: first, when a word has multiple senses, those methods compute a value of similarity for each word sense and then consider only the highest; second, while they consider the structure of the taxonomy, i.e. the relationship between taxonomic concepts, none of those measures account for the number of words belonging to those concepts.

Part II

Evaluation Methods of Word Embeddings Models

4

Setting the Stage for Word Embeddings Evaluation

In Chapter 2, we defined a word embedding model and some of the most used methods to generate word embeddings. In this Chapter, we introduce the topic of the evaluation of a word embedding model to assess its quality.

Since word embeddings are massively used in several NLP tasks, and given their high sensitivity to small changes in hyper-parameters during their generation, having a reliable measure to evaluate their goodness becomes crucial (see, e.g., [96, 37]). Indeed, the ideal evaluator should be able to analyse word embedding models from different perspectives.

Schnabel et al. [130] clarify that a *good* embedding should provide vector representations to allow the relationship between two vectors to mirror the linguistic relation between the two terms they represent.

Notably, evaluating the intrinsic quality of vector space models, as well as their impact when used as the input of specific tasks (*a.k.a.*, extrinsic quality), has a practical significance (see, e.g. [34]), as the selection of an embedding over another affects the quality of the overall process or system in which they are used. In essence, we may argue that the well-

known principle *garbage-in, garbage-out* - that denotes the data quality research field - also applies to word embeddings, as *the lower the quality of the word embeddings, the lower the effectiveness of the tasks based on them*.

4.1 Desired Properties of Embedding Evaluators

The goal of an evaluator is to compare the characteristics of different word embedding models with a quantitative and representative metric. However, finding a concrete and uniform way to evaluate these abstract characteristics is not trivial. Wang et al. [148] identify some properties that a good word embedding evaluator should aim for:

- **Testing data**: To ensure a reliable score, the testing data should have a good spread in the word space. Frequently and rarely occurring words should be included in the evaluation. Furthermore, data should be reliable, correct and objective.
- **Comprehensiveness**: Ideally, an evaluator should test for many properties of a word embedding model. This property is meaningful not only for giving a representative score but also for determining the effectiveness of an evaluator.
- **High correlation**: The score of a word model in an intrinsic evaluation task should correlate well with the model's performance in downstream natural language processing tasks. That is important for determining the effectiveness of an evaluator.
- **Efficiency**: Evaluators should be computationally efficient. Most models are created to solve computationally expensive downstream tasks. Model evaluators should be simple yet able to predict the downstream performance of a model.
- **Statistical significance**: The performance of different word embedding models concerning an evaluator should have enough statistical significance, or enough variance between score distributions, to be differentiated. That is needed in judging whether a model is better than another and helpful in determining performance rankings between models.

Existing evaluation methods fall into two major categories: extrinsic and intrinsic evaluation. In the following sections, we will present them in detail with examples for each type.

4.2 Evaluation Methods

4.2.1 Intrinsic Evaluation

Intrinsic evaluations reflect the coherence between word vectors and human judgement. These tasks typically involve a pre-selected set of query terms and semantically related target words, which we refer to as a query inventory [130]. The intrinsic term refers to the fact that it measures the structure of the vectors in the embedding space without adding external information to the model. These evaluations assess the global quality of the language representation.

Most methods of intrinsic evaluation are designed to collect judgements that are the results of conscious processes in a human brain, and such answers may be biased by certain subjective factors (for example, due to the absence of a clear definition, every person interprets word relations in their way, introducing the variability to the estimates). Following to [14] the methods for intrinsic evaluation are divided into:

- **Intrinsic conscious evaluation**: designed to compare the human assessments with those emerging from the word embedding models;
- **Intrinsic subconscious evaluation**: inspired by the classification of data collection methods in psycho-linguistic research;
- **Intrinsic thesaurus-based evaluation**: based on a comparison with knowledge bases, like taxonomies;
- **Intrinsic language-driven evaluation**: based on a comparison with data underlying in a language itself, for instance, data that could be found in speech sound signals or the frequency of occurrence of a pair of words in a corpus.

The most widely known and used metrics for intrinsic evaluation are those in the intrinsic conscious evaluation group, and there are several available datasets for these evaluation methods. The most used methods in this category are:

Similarity (or Relatedness) The performance of a model is assessed by evaluating the correlation between benchmarks constructed by asking human subjects to rate the degree of semantic similarity or relatedness between two words on a numerical scale and the cosine similarity in the word embedding model.

More formally, Spearman correlation is performed on the set of points:

$$\{(x = S_{ij}, y = cos(v_{w_i}, v_{w_j})), i \in [1, N], j \in [i+1, N]\} \tag{4.1}$$

with $(S_{ij})_{i,j \in [1,N]}$ being a matrix of pair-wise similarities obtained after compilation of several human judgements, and:

$$cos(v_{w_i}, v_{w_j}) = \frac{\langle v_{w_i}, v_{w_j} \rangle}{\| v_{w_i} \| \cdot \| v_{w_j} \|} \tag{4.2}$$

being the cosine similarity between the word embedding vectors corresponding to words w_i and w_j [139].

That is probably the most used intrinsic evaluation method (see e.g. [130, 17, 56]).

Analogy A semantic question gives an example pair (for example, brother and sister), a test word (grandson) and asks to find another word that satisfies the relation illustrated by the example pair for the test word (in this example granddaughter).

More formally, a pair of words is given (A, B) such that $\mathcal{R}(A, B)$ holds. Then, a word C is given, and the objective is to find a word D in the embedding so that $\mathcal{R}(C, D)$ holds. D is found by solving the following problems using the cosine similarity between vectors [96]:

$$3CosAdd: \underset{D \in V \setminus \{A,B,C\}}{\arg\max} \; cos(v_B - v_A + v_C, v_D) \tag{4.3}$$

$$3CosMul: \underset{D \in V \setminus \{A,B,C\}}{\arg\max} \; \frac{cos(v_D, v_B) \cdot cos(v_D, v_C)}{cos(v_D, v_A) + \varepsilon}, \quad \varepsilon = 0.001 \tag{4.4}$$

To estimate the overall performance of a dataset, we use the correct-answer accuracy.

This task was popularised by [107] and was used e.g. by [130, 17].

Concept Categorisation Given a set of concepts, the objective is to group them into categories (e.g., helicopters and motorcycles should go to the vehicle class, and dogs and elephants in the mammal class). It can be considered an unsupervised clustering task.

More formally, a categorisation dataset D is defined by a set of classes $\mathcal{C} = (C_i)_{i\in[1,N]}$, and a set of words $W = (w_j)_{j\in[1,K]}$, such that each word belongs to a specific class: $\mathcal{D} = \{(w, C(w)), w \in W\}$. The goal is to create a set of clusters $\mathcal{C}^r = (C_i^r)_{i\in[1,N]}$ with the K word vectors. $\mathcal{C}$ and $\mathcal{C}^r$ are then compared using purity [139]:

$$Purity(\mathcal{C}, \mathcal{C}^r) = \frac{1}{K}\sum_{i=1}^{N} \max_{j\in[1,N]} |C_i \cap C_j^r| \tag{4.5}$$

It was used e.g. by [130, 17].

Outlier Word Detection The task is to identify a semantically anomalous word in an already formed cluster (for example, for a set banana, lemon, book, orange the word book is the outlier since it is not a fruit).

More formally, we can take a set of words $W = \{w_1, w_2, \ldots, w_{n+1}\}$ where there is one outlier, and we take a compactness score of word w as:

$$c(w) = \frac{1}{n(n-1)} \sum_{w_i \in W\setminus w} \sum_{w_j \in W\setminus w, w_j \neq w_i} sim(w_i, w_j) \tag{4.6}$$

The outlier is the word with the lowest compactness score, and to estimate the overall performance on a dataset, we use the correct-answer accuracy [33].

The method was introduced by [33], even though a similar strategy, called coherence, was previously introduced in [130].

Synonym Detection It consists of multiple-choice questions that pair a target term with four synonym candidates (e.g. for the target levied, one must choose between imposed (correct), believed, requested, and correlated). For each candidate vector, the cosine similarity with the target is computed, and the candidate with the largest cosine is chosen. The performance is evaluated in terms of correct-answer accuracy [17].

It was used, e.g., by [17].

Selectional Preferences (or Thematic Fit) It uses datasets containing verb-noun pairs rated by subjects for how typical it is to have the noun as a subject or object of the verb. For each verb, the aim is to select the twenty nouns most strongly associated with it

as subjects or objects and to average the vectors of these nouns to obtain a *prototype* vector for the relevant argument slot.

Then, the aim is to measure the cosine of the vector for a target noun with the relevant prototype vector (e.g., the cosine of people with the eating subject prototype vector). Systems are evaluated by Spearman correlation of these cosines with the averaged human typicality ratings[17].

Then, the cosine similarity of a target noun with the prototype vector is computed (e.g., the cosine of people with the eating subject prototype vector). It was used e.g. by [130, 17].

Since `TaxoVec`, the evaluation method we propose in Chapter 5, is similar to the *intrinsic thesaurus-based evaluation metrics*, we also present the most used methods in this category in the following paragraphs.

Thesaurus Vectors Evaluation Method (QVEC) It is based on the idea that word embeddings can be evaluated with the vectors of the inverted index of a collection of documents (the thesaurus vectors) in which each is responsible for a certain category of human knowledge, like super-senses in WordNet (e.g. food or animal). The dimensionality of the thesaurus vectors is equal to the size of the collection, and each component reports the number of occurrences of the word in a certain document. The gold standard is represented by the thesaurus vectors. This method was introduced by [141].

Semantic Networks Evaluation Method It uses manually constructed knowledge graphs (semantic networks) as a gold standard. In semantic networks, the words are organised in a graph by their semantic distinctive features based on the judgements of teams of professional linguists. Semantic networks also feature a measure of similarity for word pairs based on the shortest path in a graph, which can be compared with the similarity measure of the same pair calculated by word embeddings [2]. The most well-known example of such a semantic network is WordNet; another popular semantic network is DBpedia.

4.2.2 Extrinsic Evaluation

Methods of extrinsic evaluation are based on the ability of word embeddings to be used as the feature vectors of supervised machine learning algorithms, and the performance of the supervised model (being measured on a dataset for NLP task) functions as a measure of word embedding quality. Word embeddings probably could be used in almost any NLP task, and thus any of them could be considered an evaluation method.

If word embeddings are supposed to be used only to resolve a specific downstream task, the evaluation of the performance of a supervised model on this task will give an adequate score of word embeddings performance. But extrinsic evaluation fails if the embeddings that one wants to evaluate are needed in a wide range of different tasks since various downstream tasks differ very much, and word embeddings performance scores in various downstream tasks do not correlate between themselves [14].

In the following, we present the most used metrics for extrinsic evaluation:

POS Tagging POS tagging aims to assign tags to each input token with its Part Of Speech type - like noun, verb, adverb, or conjunction. Due to the availability of labelled corpora, many methods can complete this task by either learning probability distribution through linguistic properties or statistical machine learning.

Chunking The goal of chunking, also called shallow parsing, is to label segments of a sentence with syntactic constituents. Each word is first assigned with one tag indicating its properties, such as noun or verb phrases. It is then used to syntactically group words into correlated phrases. Compared with POS, chunking provides more clues about the structure of the sentence or phrases in the sentence.

Named-Entity Recognition (NER) The NER task is widely used in natural language processing, and it focuses on recognising inside sentences entity classes such as names (e.g. person, location, or organisation) and numeric expressions (e.g. time and percentage). Like the POS tagging task, NER systems use both linguistic grammar-based techniques and statistical models, with the former requiring efforts on experienced linguists and the latter requiring a large amount of human-labelled data for training.

Sentiment Analysis (SA) Sentiment analysis is a sentence-level text classification task. Usually, a text fragment is marked with a binary - or multi-level - label representing the positiveness or negativeness of the text's sentiment. Traditional methods focus more on human-labelled sentence structures, while with the development of machine learning, more statistical and data-driven approaches have been proposed to deal with the sentiment analysis task.

4.3 Consistency Study of Extrinsic and Intrinsic Evaluators

Wang et al. [148] conducted a consistency study of extrinsic and intrinsic evaluators via a correlation analysis. Figure 4.1 shows the Pearson correlation of each intrinsic and extrinsic evaluation pair of 16 different word embedding models. For example, the entry of the first row and the first column is the Pearson correlation value of 16 evaluation data pairs of WS-353 (an intrinsic evaluator) and POS (an extrinsic evaluator).

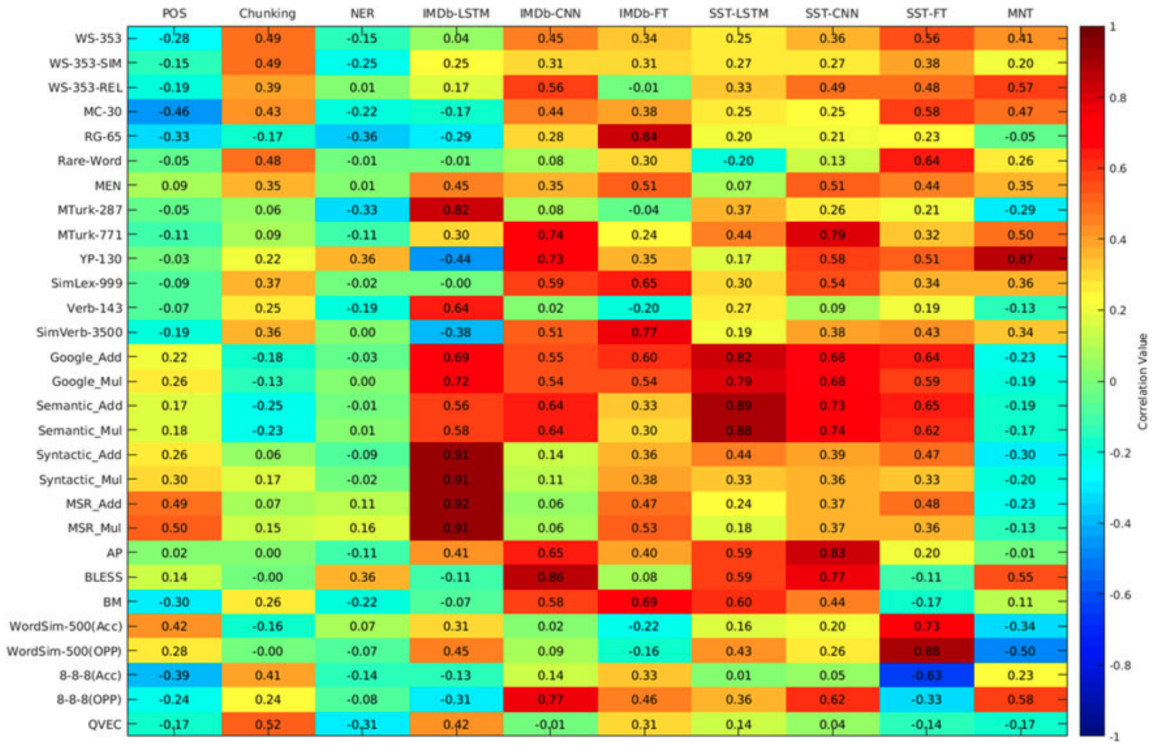


Figure 4.1 Pearson's correlation between intrinsic and extrinsic evaluator, where the x-axis shows extrinsic evaluators while the y-axis indicates intrinsic evaluators. The warm indicates the positive correlation while the cool colour indicates the negative correlation [148].

The top 13 rows in Figure 4.1 correspond to 13 word similarity evaluation datasets; the word analogy results are shown from the 14th row to the 21st row; then there are three datasets (i.e. AP, BLESS, and BM) for concept categorisation two (WordSim-500 and 8-8-8) are used for outlier detection, and the last one is for QVEC.

For the similarity datasets, the authors found that larger datasets tend to give more reliable and consistent evaluation results; they also found that word analogy provides the most reliable correlation results and has the highest correlation with the sentiment analysis task. Moreover, the authors found that the outlier detection task does not seem to be a useful evaluation method, but they expect that, with larger and more reliable datasets available, the performance should be better. They concluded that word similarity, word analogy, and concept categorisation are the most effective intrinsic evaluators and that larger datasets tend to give better and more reliable results. Intrinsic evaluators may perform very differently for different downstream tasks, and therefore they suggest using those jointly[148].

4.4 Related Work on Word Embedding Evaluation

There is extensive literature on intrinsic evaluation methods for word vector models. Baroni et al. [17] were the first to systematically compare word embeddings against the count-vector-based distributional word vectors on various query tasks. In their paper, they perform an extensive evaluation on a wide range of lexical and semantics tasks across many parameter settings. Most of those tasks were already widely used to test and compare distributional semantic models.

Then, Schnabel et al. [130] provided a comprehensive study covering a wide range of evaluation criteria and popular embedding techniques was conducted. The authors analyse existing evaluation methods and introduce novel ones, discussing the relative strengths and limitations. In their article, they clarify that various evaluations result in different orderings of the embeddings, raising questions on the common assumption that there can be a single optimal vector representation.

Wang et al. [148] performed an extensive evaluation on many word embedding models for language processing applications. They report experimental results of intrinsic and extrinsic evaluators on six-word embedding models, showing that different evaluators focus on distinct

aspects of word models, and some are more correlated with natural language processing tasks. Torregrossa et al. [140] provided a similar study focused on the correlation of word embedding evaluation metrics on various datasets.

Bakarov [14] presented an extensive overview of the field of word embeddings evaluation, proposing a new kind of intrinsic word embeddings evaluation methods, considering both widely used (and at the same time widely-criticised) and less mainstream.

Torregrossa et al. [139] provide a survey that focuses on the algorithms and models used to compute word embeddings and on their methods of evaluation. They also supply a comparison of those algorithms and methods, highlighting open problems and research paths; moreover, they contribute a compilation of popular evaluation metrics and datasets.

Lai et al. [92] proposed a measure that normalises the performances on all tasks to the same scale, thereby simplifying the analysis across multiple tasks. Their indicator is the Performance Gain Ratio (PGR): the Performance Gain is the performance gain offered by the embedding compared to a random model, while the PGR is the ratio of the Performance Gain of a model compared to another one.

Faruqui and Dyer [56] provided a website that allowed the automatic evaluation of embeddings. They created an online application that enables users to (i) access a suite of word similarity evaluation benchmarks, (ii) evaluate user-computed word vectors, (iii) visualise word vectors in a two-dimensional space, and (iv) upload user vectors for exhaustive offline evaluation. The online application allows the users to evaluate the embeddings using the benchmarks, but it does not provide an overall evaluation based on those benchmarks across different tasks. Unfortunately, the website is not maintained anymore.

In the following Chapters, we will propose two frameworks for the evaluation of word embeddings:

- In Chapter 5, we propose `TaxoVec` - a framework to select taxonomy-aware word embeddings, based on the `HSS` - a measure of semantic similarity between taxonomic words we developed.
- In Chapter 6, we propose the `vec2best` tool, a unified approach to several state-of-the-art intrinsic evaluation tasks over different benchmarks, which produces a comprehensive measure of evaluation for each model called the *PCE (Principal Component Evaluation)*.

5

Embeddings Evaluation Using a Novel Measure of Semantic Similarity

In Chapter 4, we first provided an overview of the prevailing methods utilised for assessing word embedding models; in this Chapter, we introduce a novel metric and methodology for the evaluation of word embeddings. To be more specific, we propose (i) the `HSS`, as a measure of semantic similarity between taxonomic words, (ii) a python package[1], fully deployed and available to the whole community, which allows to compute the `HSS` (and all the other semantic similarity metrics considered) between any pair of WordNet nouns, and (iii) `TaxoVec`, a framework to select taxonomy-aware word embeddings[2].

We perform an extensive set of intrinsic and extrinsic evaluation tasks on well-known benchmarks to evaluate `TaxoVec` against the state-of-the-art measures of taxonomic semantic similarity and methods for embedding evaluation.

[1] https://pypi.org/project/TaxoSS

[2] Some results of this Chapter were published in [68].

5.1 Motivation

In recent years, we have witnessed a clear dominance of learning-based methods in NLP applications [114]. Despite sub-symbolic AI (i.e., machine learning) has shown to outperform symbolic AI (i.e., knowledge-based AI) in retrieving linguistic patterns from a large amount of data and in making predictions, those methods still suffer from dependency issues, i.e. they require large amounts of training data and are domain dependent [36]. This issue is particularly severe in domains where it is challenging to retrieve labelled data, and there is extensive use of jargon and linguistic features which sub-symbolic AI struggles to learn, like rhetoric, irrealis moods, metaphors, etc. [49, 155]. For those reasons, enhancing distributional models with lexical knowledge brings noteworthy benefits to their use as input features in several downstream machine-learning tasks.

On the other side, lexical taxonomies are manually built, and their creation and update are time-consuming, error-prone, require domain-specific knowledge, and usually have low coverage [62]. The use of taxonomy-aware embeddings can be beneficial to automatically infer semantic information from domain-specific text corpora to build, update or maintain lexical taxonomies [69, 104], as will be further discussed in the third part of this thesis.

The evaluation method we propose is similar to the *intrinsic thesaurus-based evaluation metrics* as it uses an expert-constructed taxonomy to evaluate the word vectors as an intrinsic metric [17, 78]. Typically, these approaches evaluate embeddings by their correlation with manually crafted lexical resources, like expert rating of similarity or relatedness between hierarchical elements. However, those resources are usually limited and hard to create and maintain. Furthermore, human similarity judgements evaluate ex novo the semantic relatedness between taxonomies, but the taxonomy's structure already encodes information about the relations between its elements. In this work, instead, we exploit the information encoded in an existing taxonomy to build a benchmark for the evaluation of word embeddings.

Selecting a word vector model that represents and preserves taxonomic similarity relations would allow us to generate a unified representation of knowledge-based and data-driven lexical features and would enable several NLP applications. Some of them are related to the maintenance and update of the taxonomy itself, like taxonomy refinement and enrichment (see Chapter 8) or taxonomy alignment (see Chapter 9). Other applications are downstream tasks

that rely on underlying structured knowledge representation, like recommendations for online retailers [160], query understanding for search engines [79], and text understanding [152], to cite a few of them. For this reason, we develop `TaxoVec`, a framework for embedding selection driven by a measure of semantic similarity between taxonomic elements (`HSS`).

Semantic similarity represents a particular case of semantic relatedness [124], considering only the co-hyponymy and synonymy relations. For instance, the words *cat* and *tiger* are more similar than the words *jungle* and *tiger*, while the latter pair seems to be more related. The semantic similarity strongly depends on the context. For this reason, it is valuable to find an automated way to compute the similarity between words in a domain-dependent taxonomy. In Chapter 3, we presented different approaches for measuring semantic similarity in a taxonomy. Despite all of them being relevant, they have two main drawbacks: first, when a word has multiple senses, those methods compute a value of similarity for each word sense and then consider only the highest; second, while they consider the structure of the taxonomy, thus the relationship between taxonomic concepts, none of those measures accounts for the number of words belonging to those concepts. To overcome these limitations, we developed the `HSS`, a measure that has proven to be valuable in several applications, like taxonomy enrichment [69, 70], refinement [105], alignment [67], and job-skill mismatch analysis in the field of labour market [71].

5.2 Methodology

In this section, we will:

- Define the `HSS`, a measure of similarity within a semantic hierarchy, which will serve as a basis for the embeddings evaluation;
- Present *TaxoSS*, a tool for computing Semantic Similarity using the `HSS` and other state-of-the-art measures (see Chapter 3);
- Present `TaxoVec`, the framework for the evaluation of word embedding models using a measure of similarity within a semantic hierarchy (like the `HSS`). The `TaxoVec`'s steps are shown in Figure 5.1.

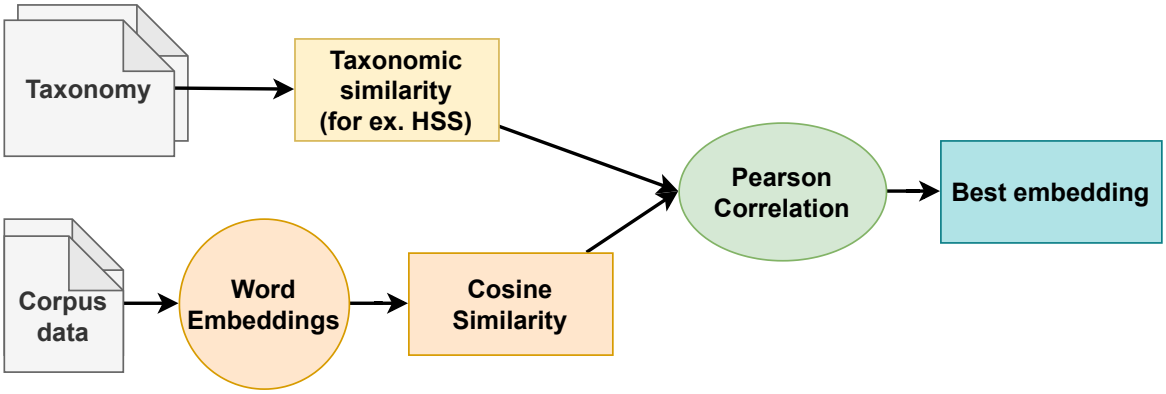


Figure 5.1 `TaxoVec`*'s workflow.*

5.2.1 Hierarchical Semantic Similarity (HSS)

To compute the Hierarchical Semantic Similarity (HSS) - similarly to [132] - we compute $\hat{p}(c)$ using an intrinsic measure, exploiting the structure of the taxonomy instead of an external corpus. However, the HSS, differently from [132], which uses only the number of taxonomic concepts, considers also the entities of the taxonomy:

$$\hat{p}(c) = \frac{N_c}{N} \tag{5.1}$$

where N is the cardinality, i.e., the number of entities (words) of the taxonomy and N_c is the sum of the cardinality of the concept c with the cardinality of all its hyponyms. Note that $\hat{p}(c)$ is monotonic and increases with granularity, thus respects our definition of p (see Sec. 3.2 of Chapter 3).

Now, given two words w_1 and w_2, Resnik defines $c_1 \in s(w_1)$ and $c_2 \in s(w_2)$ to be all the concepts containing w_1 and w_2 respectively, i.e. the *senses* of w_1 and w_2. Therefore, there are $|s(w_1)| \times |s(w_2)|$ possible combinations of their word senses, where $|s(w_1)|$ and $|s(w_2)|$ are the cardinality of $s(w_1)$ and $s(w_2)$ respectively. We can now define $\mathcal{L}$ as the set of all the lowest common ancestors for all the combinations of $c_1 \in s(w_1), c_2 \in s(w_2)$.

The hierarchical semantic similarity between the words w_1 and w_2 can therefore be defined as:

$$sim_{\text{HSS}}(w_1, w_2) = \sum_{\ell \in \mathcal{L}} \hat{p}\left(\ell = LCA \mid w_1, w_2\right) \times IC(LCA) \tag{5.2}$$

Where $\hat{p}(\ell = LCA \mid w_1, w_2)$ is the probability of *LCA* being the lowest common ancestor of w_1, w_2, and can be computed as follows applying the Bayes theorem:

$$\hat{p}(\ell = LCA \mid w_1, w_2) = \frac{\hat{p}(w_1, w_2 \mid \ell = LCA)\,\hat{p}(LCA)}{\hat{p}(w_1, w_2)} \tag{5.3}$$

We define N_ℓ as the cardinality of ℓ and all its descendants.

Now we can rewrite the numerator of Eq. 5.3 as:

$$\hat{p}(w_1, w_2 \mid \ell = LCA)\,\hat{p}(\ell = LCA) = \frac{S_{<w_1,w_2>\in\ell}}{|descendants(\ell)|^2} \times \frac{N_\ell}{N}. \tag{5.4}$$

Where the first leg of the *right-hand side* is the class conditional probability of the pair $< w_1, w_2 >$ having ℓ as the lowest common ancestor and the second one is the marginal probability of the class ℓ. The term $|descendants(\ell)|$ represents the number of sub-concepts of ℓ. Since we could have at most one-word sense w_i for each concept c, $|descendants(\ell)|^2$ represents the maximum number of combinations of word senses $< w_1, w_2 >$ which have ℓ as lowest common ancestor. $S_{<w_1,w_2>\in LCA}$ is the number of pairs of senses of word w_1 and w_2 which have *LCA* as lower common ancestor.

The denominator of Equation 5.3 can be written accordingly as:

$$\hat{p}(w_1, w_2) = \sum_{k \in \mathcal{L}} \frac{S_{<w_1,w_2>\in k}}{|descendants(k)|^2} \tag{5.5}$$

5.2.2 *TaxoSS*: a Tool for Computing Semantic Similarity

Semantic similarity can be useful for a vast number of tasks, with embedding selection being the one we are focusing on here. For this reason, we decided to implement the `HSS` and all the other automatic semantic similarity measures considered in Sec. 3.2 of Chapter 3 as a fully-fledged python package. That is going to facilitate the user who wants to use it. This library, called *TaxoSS* (Taxonomic Semantic Similarity), allows computing taxonomic-based similarity (using WordNet 3.0) as well as corpus-based similarity measures. For the latter, there is a default measure based on the English Wikipedia dump of the year 2008, but the user can also use a different corpus.

The python library *TaxoSS* that we created allows the user to easily compute semantic similarity between concepts using eight different measures: HSS, WUP, LC, Shortest Path, Resnik, Jiang-Conrath, Lin, and Seco.

Fig. 5.2 and 5.3 show the use of the package for the computation of the semantic similarity between two words using different metrics. Fig. 5.3 also shows how the user can use their corpus to compute the similarity through corpus-based similarity measures.

```
from TaxoSS.functions import semantic_similarity
semantic_similarity('brother', 'sister', 'hss')

3.353513521371089
```

Figure 5.2 An example of the use of the semantic similarity *function with the HSS metric.*

```
from TaxoSS.functions import semantic_similarity
semantic_similarity('cat', 'dog', 'resnik')

6.169410755220327

from TaxoSS.calculate_IC import calculate_IC
calculate_IC('data/corpus_test.csv', 'venv/lib/python3.6/site-packages/TaxoSS/data/test_IC.csv')
semantic_similarity('cat', 'dog', 'resnik', 'venv/lib/python3.6/site-packages/TaxoSS/data/test_IC.csv')

3.5077209766856137
```

Figure 5.3 An example of the use of the semantic similarity *function with Resnik metric and the use of an ad hoc Information Content file created through a corpus of choice.*

5.2.3 TaxoVec: Embeddings Evaluation and Selection of the Best Embedding

Following the previous literature on intrinsic word embedding evaluation (see Chapter 4), which correlates the cosine similarity between pairs of word vectors with human scores of relatedness/similarity, we assess the goodness of a vector model by the Pearson correlation coefficient between the cosine similarity of pairs of word vectors and their taxonomic semantic similarity.

The taxonomic semantic similarity can be measured with all the metrics presented in Sec. 3.2 of Chapter 3 or using the HSS (see Section 5.2.1).

In other words, to obtain the embedding evaluation using a measure of semantic similarity, we generate a variety of vector representations of a large text corpus, and we select the one that

better represents the taxonomy according to different measures of semantic similarity. That is, we want the similarity between word vectors to reflect as much as possible the semantic similarity between words in the taxonomy, so we select the embedding that maximises the correlation between cosine similarity and semantic similarity.

5.3 Experimental Results

In this experimental section, we select the best word embedding model as the one that maximises the correlation between cosine similarity and semantic similarity as it is computed by HSS and WUP (see Section 3.2), and as the one that maximises the correlation between cosine similarity and the similarity assessed by the evaluators in the benchmarks MEN and SimLex999 (see Section 6.4.2). To evaluate the effectiveness of the HSS for the selection of word embeddings, we perform intrinsic and extrinsic evaluations to compare the embedding model selected through HSS, WUP, MEN, and SimLex999.

Our experiments rely on a lexical taxonomy and a corpus:

- **Taxonomy:** WordNet [109] provides a structured hierarchy of meanings (senses) and synsets (a collection of words belonging to a specific context). We used the implementation of WordNet inside the NLTK library [22] while calculating the required information using NLTK's native functions (e.g., calculating the lowest common ancestor) and custom functions (e.g., calculation of cardinality).
- **Corpus:** English Wikipedia dump of the year 2008. The main reason for not choosing a more recent update is that the last release of WordNet 3.0 (the version we used for our experiments) was from 2006, making the use of a newer Wikipedia dump version unnecessary. The dump used already includes the pre-processed data version that we used without performing further cleaning.

We generate 80 different embedding models with fastText. Among them, we select the four that better correlate their cosine similarity with the HSS, WUP, MEN, and SimLex999. To evaluate these four models and compare the semantic similarity measures used to select them, we use their vectors as input features in four downstream NLP tasks. The four NLP tasks are, in order of presentation: categorisation, sentiment classification, hypernym detection, and synonym detection.

5.3.1 Generation of the Embeddings

We trained our vector models with the fastText library using both *skipgram* and *CBOW*. We tested the following parameters:

- Five values of embeddings sizes: 25, 50, 100, 250, and 500;
- Four for the number of epochs: 5, 10, 20, and 50;
- Two for the learning rate: 0.05 and 0.1.

We considered sub-words with 5 to 6 letters while setting the minimum word count as 100, running on an Intel Core i7 CPU with 32 GB RAM. The best embeddings chosen by each measure for each dataset are reported in Tab. 5.1.

Table 5.1 Best embeddings for each measure.

	Algorithm	**Size**	**Epoch**	**Learning rate**
HSS (ours)	SG	500	5	0.05
WUP [153]	CBOW	50	10	0.01
MEN [32]	SG	250	10	0.05
SimLex999 [78]	CBOW	500	50	0.01

Given that with HSS and WUP, we can compute the similarity for each pair of terms in the taxonomy, for this evaluation we have $(n \times (n-1))/2$ pairs of terms for each vector model, where n is the number of words present both in the taxonomy and the text corpus. In our case, the number of common words is 53,451, for a total of 1,428,477,975 possible word pairs for each model, which would make the computation intractable. To reduce the number of samples, we start from 0 pairs and, following [131], we increase the sample size until the Pearson correlation stabilises. The process of choosing the number of pairs is detailed in the following paragraph.

5.3.2 Choosing the Number of Pairs

To define the minimum number of pairs required for our experiments, after randomly generating 100k pairs presented both in WordNet and our corpus vocabulary, we recursively generate samples of pairs while increasing the number of samples by 100 in each step. Fig. 5.4 shows the Pearson correlation of the HSS score and cosine similarity of each paired sample, while the orange line indicates the rolling average of 100. To define the Point of Stability (PoS)

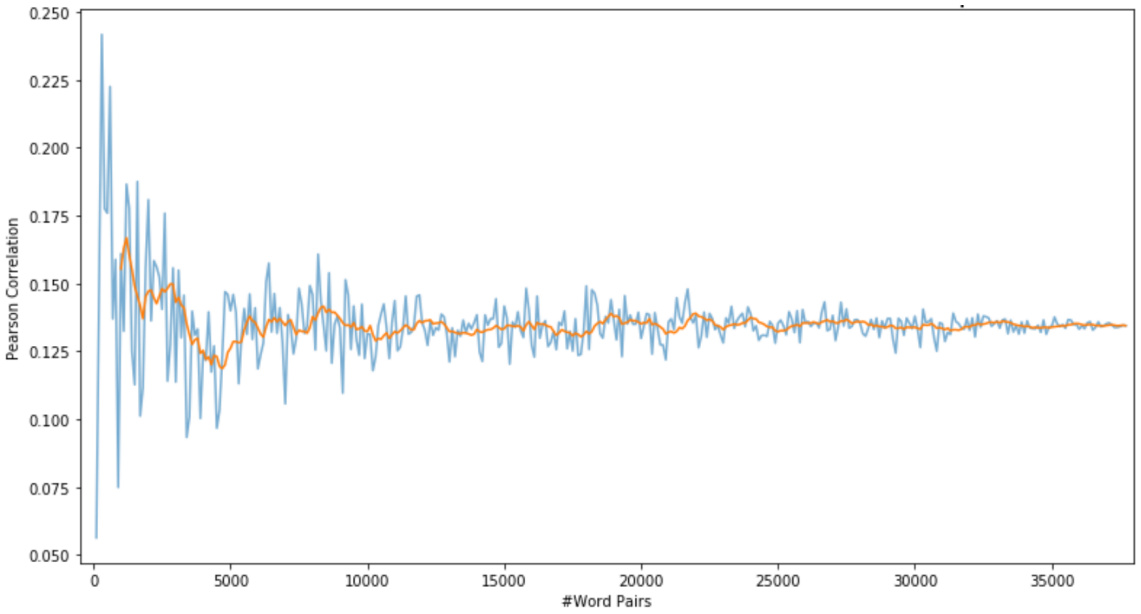


Figure 5.4 Variation in Pearson Correlation of HSS score and cosine similarity vs. the number of word pairs. The orange line indicates the rolling average of 100.

(i.e., a point from which the correlation remains within the POS), we considered a Corridor of Stability (CoS) of ± 0.01. Although the rule of thumb proposed by [43] and the work of [131] suggest a larger CoS (between 0.05 and 0.1) due to difficulties in achieving a tighter CoS mainly caused by the cost of the additional samples. Since such a cost is not relevant in our case, we decided to consider COS as ± 0.01, which results in a PoS of 35,000.

Comments on the Best Embedding To better clarify the matter, using the *ap* [7] dataset, in Fig.5.5, we provide a scatter plot produced over the best embedding model - as emerges from Tab. 5.1 - generated with UMAP[3]. We chose twenty random records for the first ten categories. Each icon and colour is assigned to one category, demonstrating how well the clusters are separated from each other. Analysing the scatter plot, it can be observed that:

- Categories that are conceptually more distant concerning the other categories show a clearer separation, for instance ◆ ***chemical element*** and × ***monetary unit***, while categories semantically close together have less clear boundaries, such as ⬢ ***legal documents*** and ★ ***assets***;

[3]Uniform Manifold Approximation and Projection (UMAP) is a dimension reduction technique that can be used for visualisation similarly to t-SNE, but also for general non-linear dimension reduction

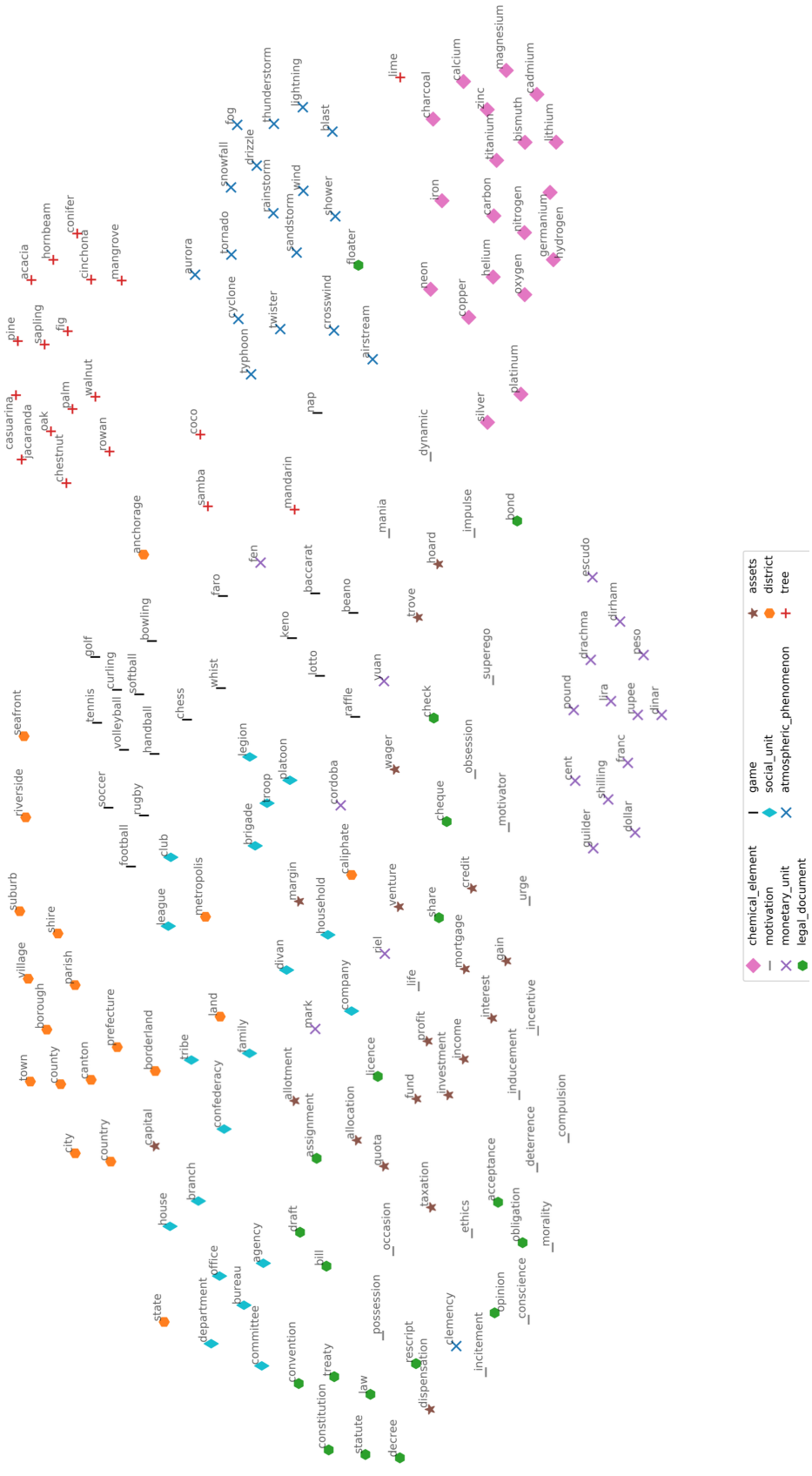


*Figure 5.5 UMAP plot of the **best** word-embedding model resulting from Tab. 5.1, that is algorithm=skipgram, size=500, epochs=5, and learning rate = 0.05. Each icon is assigned to one category in ap [7].*

- Some cases are on the borderline of two or more classes. While such cases may seem to show model weakness, such observations can be explained based on the nature of the non-contextual embeddings since by definition they cannot represent words with multiple meanings. For example, in the scatter plot mentioned above, the word ***capital***, which belongs to ★ ***assets*** in WordNet, is on the border of ♦ ***social unit*** and ⬢ ***district***. That can be explained by using the Wikipedia corpus (i.e., a generic corpus) for generating the embeddings.

Below, we describe the four downstream NLP tasks performed to evaluate the embedding selection procedure. For each task, we comment on the results.

5.3.3 Evaluation on Natural Language Processing Tasks

Categorisation

Following [17], we cluster the vectors from each selected embedding, applying the k-means algorithm from the scikit-learn library [116], with the default parameters. In the next step, the purity of each cluster is calculated. The number of clusters is equal to the number of classes in each dataset. To account for the variation in results we compute the average value of 50 iterations for each pair.

The datasets used in these experiments are AP [7], containing 403 concepts organised into 21 categories, BM [20], including 4,668 concepts belonging to 56 categories, and ESSLLI [18], from the ESSLLI 2008 Distributional Semantic Workshop shared-task set, containing 45 concepts divided into nine categories.

The purity values of clusters are reported in Tab. 5.2, where a purity close to 1 shows that the cluster is well reproduced, while a purity close to 0 indicates poor cluster quality. These results show that the embedding chosen by our similarity measure outperforms the other three embeddings, chosen by WUP, MEN, and SimLex999 measure when applied on three well-known categorisation datasets used by [17].

Sentiment Classification

We carry out the sentiment classification task on three user review datasets: Binary and Multi-class Amazon Review and Binary Movie Review.

Table 5.2 Categorisation: Cluster purity obtained from each embedding and dataset.

	AP [7]	**BM** [20]	**ESSLLI** [18]
HSS (ours)	**0.75**	**0.62**	**0.81**
WUP [153]	0.68	0.43	0.73
MEN [32]	0.71	0.58	0.77
SimLex999 [78]	0.71	0.49	0.78

Replicating the experiment done in [130], we perform binary sentiment classification on the dataset from [101]. This dataset contains 50K movie reviews divided equally between binary labels. Utilising the embeddings as the features of a logistic regression [55], we compute a linear combination of embeddings, weighted by the word count in each review. We use the Scikit-learn library [116] to apply the mentioned regression and calculate the accuracy values for each selected embedding by performing a 10-fold cross-validation. The results of this task (classification accuracy) can be seen in Tab. 5.3.

Using the binary and the multi-class Amazon cellphone review datasets[4] we perform both binary and multi-class sentiment classification. These datasets contain $26,845$ and $29,988$ reviews labelled 0 or 1 for the binary dataset and from 1 (absolutely negative) to 5 (absolutely positive) for the multi-class dataset. In both cases, we down-sample the dataset based on the minority label. Tab. 5.3 shows the mean and the standard deviation of the performance metrics that result from 10-fold cross-validation. Our measure can outperform WUP, MEN, and SimLex999 benchmarks. The exception is the multi-class dataset, in which HSS produces results similar to those of SimLex999.

Hypernym Detection

For this task, we employ the BATS benchmark dataset [73], which contains 99,200 questions in 40 morphological and semantic categories. In particular, we use three lexicography categories, namely, *Hypernyms* (Animals, Miscellaneous) and *Hyponyms* (Miscellaneous).

To generate hypernym-hyponym pairs (corresponding to the positive pairs), we extract all the possible pairs, deduplicate them, and remove all the pairs with at least one word from the embedding vocabulary. The process leads to 1,129 pairs. To generate the negative pairs, we

[4] https://jmcauley.ucsd.edu/data/amazon/

Table 5.3 Sentiment Classification.

	Accuracy	**Precision**	**Recall**	**F1**
		Multi-class Amazon Review		
HSS (ours)	**0.4** $\pm$ 0.04	**0.4** $\pm$ 0.04	**0.41** $\pm$ 0.04	**0.4** $\pm$ 0.04
WUP [153]	0.34 $\pm$ 0.06	0.33 $\pm$ 0.04	0.34 $\pm$ 0.06	0.31 $\pm$ 0.05
MEN [32]	0.4 $\pm$ 0.04	0.39 $\pm$ 0.04	0.4 $\pm$ 0.04	0.39 $\pm$ 0.04
SimLex999 [78]	**0.41** $\pm$ 0.02	**0.4** $\pm$ 0.02	**0.41** $\pm$ 0.02	**0.4** $\pm$ 0.02
		Binary Amazon Review		
HSS (ours)	**0.82** $\pm$ 0.02	**0.81** $\pm$ 0.03	**0.84** $\pm$ 0.03	**0.82** $\pm$ 0.02
WUP [153]	0.72 $\pm$ 0.03	0.71 $\pm$ 0.04	0.74 $\pm$ 0.04	0.72 $\pm$ 0.03
MEN [32]	0.81 $\pm$ 0.02	0.79 $\pm$ 0.03	0.83 $\pm$ 0.02	0.81 $\pm$ 0.02
SimLex999 [78]	0.8 $\pm$ 0.04	0.79 $\pm$ 0.04	0.82 $\pm$ 0.05	0.8 $\pm$ 0.04
		Binary Movie Review		
HSS (ours)	**0.84** $\pm$ 0.01	**0.84** $\pm$ 0.01	**0.84** $\pm$ 0.02	**0.84** $\pm$ 0.01
WUP [153]	0.72 $\pm$ 0.02	0.72 $\pm$ 0.02	0.72 $\pm$ 0.02	0.72 $\pm$ 0.02
MEN [32]	0.82 $\pm$ 0.01	0.82 $\pm$ 0.01	0.82 $\pm$ 0.02	0.82 $\pm$ 0.01
SimLex999 [78]	0.83 $\pm$ 0.01	0.83 $\pm$ 0.01	**0.84** $\pm$ 0.02	0.83 $\pm$ 0.01

use the aforementioned categories to randomly select words and form pairs, arriving at 1,129 pairs that do not have a hypernym-hyponym relationship. Finally, by subtracting the vectors of pair words, we create a single vector, and we use it as the feature for training the classifier. To compensate for the potential effect of the randomly generated pairs on the results, we repeat the process ten times, each time using a logistic regression model to classify the pairs.

Tab. 5.4 shows the mean and standard deviation of the outcomes for chosen embeddings. The `HSS` outperforms the other benchmarks except for the MEN dataset, which achieves the same precision as our method.

Table 5.4 Hypernym detection.

	Accuracy	**Precision**	**Recall**	**F1**
HSS (ours)	**0.72** $\pm$ 0.013	**0.72** $\pm$ 0.012	**0.75** $\pm$ 0.017	**0.73** $\pm$ 0.013
WUP [153]	0.68 $\pm$ 0.015	0.71 $\pm$ 0.02	0.65 $\pm$ 0.014	0.68 $\pm$ 0.013
MEN [32]	0.71 $\pm$ 0.014	**0.72** $\pm$ 0.017	0.72 $\pm$ 0.015	0.72 $\pm$ 0.012
SimLex999 [78]	0.69 $\pm$ 0.023	0.68 $\pm$ 0.251	0.73 $\pm$ 0.017	0.70 $\pm$ 0.019

Synonym Detection

As for the previous task, we used the BATS benchmark to generate the training data for the logistic regression classifier. To create positive pairs (i.e., pairs with synonym relationships), first, we generate all the possible combinations of 2 for each entry in *Synonym-exact* and *Synonym-intensity* files. Then, we combine them with the pair made from the synonymous words in the *Antonym-gradable* file, which results in 4,663 pairs that we reduced to 4,011 pairs after de-duplication. Similar to the method described in the previous task, we generate negative pairs by iterating through the vocabulary ten times.

As reported in Tab. 5.5, despite close results to the SimLex999 dataset, the embedding chosen by the `HSS` carries out a better synonym classification than the other methods.

Table 5.5 Synonym detection.

	Accuracy	**Precision**	**Recall**	**F1**
`HSS` (ours)	**0.616** $\pm$ 0.006	**0.609** $\pm$ 0.007	**0.624** $\pm$ 0.012	**0.617** $\pm$ 0.006
WUP [153]	0.54 $\pm$ 0.007	0.534 $\pm$ 0.007	0.537 $\pm$ 0.014	0.536 $\pm$ 0.007
MEN [32]	0.586 $\pm$ 0.006	0.58 $\pm$ 0.007	0.586 $\pm$ 0.008	0.583 $\pm$ 0.006
SimLex999 [78]	0.612 $\pm$ 0.007	0.605 $\pm$ 0.008	0.618 $\pm$ 0.004	0.611 $\pm$ 0.005

6

A Unified Framework for Intrinsic Evaluation of Word-Embedding Algorithms

In Chapter 4, we provided an overview of the most common methods for assessing word embedding models, and in Chapter 5 we introduced a framework to select taxonomy-aware word embeddings leveraging a measure of taxonomic semantic similarity (the `HSS`).

In this Chapter, we propose the `vec2best` tool [10], a unified approach to several state-of-the-art intrinsic evaluation tasks over different benchmarks, which provides the user with an extensive evaluation of word embedding models. `vec2best` represents an approach to evaluate word embeddings trained with different methods and hyper-parameters on a set of tasks from the literature and produces a comprehensive measure of evaluation for each model called the *PCE (Principal Component Evaluation)*[1].

[1] Some results of this Chapter are in press at Cognitive Computation Journal [11].

6.1 Motivation

Evaluating the intrinsic quality of vector space models, as well as their impact when used as the input of specific tasks (*a.k.a.*, extrinsic quality), has a practical significance, see, e.g. [34], as the selection of an embedding over another affects the quality of the overall process or system in which they are used. In essence, we may argue that the well-known principle *garbage-in, garbage-out* - that denotes the data quality research field - also applies to word embeddings, as *the lower the quality of the word embeddings, the lower the effectiveness of the tasks based on them*.

The state-of-the-art approaches for word embedding evaluation can be divided into two major classes, as seen in Chapter 4: intrinsic and extrinsic evaluation.

Intrinsic evaluation of word embeddings directly test for syntactic or semantic relationships between words [107, 17]. Those tasks typically involve a pre-selected set of query terms and semantically related target words, with human judgements on word relations (i.e., absolute intrinsic evaluation).

Methods of extrinsic evaluation are based on the ability of word embeddings to be used as the input features of machine learning algorithms; the performance of the downstream model acts as a measure of word embeddings quality [14]. Extrinsic evaluators are more computationally expensive and may not be directly applicable. Moreover, Schnabel et al. [130] state that extrinsic evaluations, although valid in highlighting specific aspects of embedding performance, should not be used as a proxy for generic quality.

In essence, the *contributions* of `vec2best` are:

1. It represents an approach to evaluate word embeddings trained with different methods and hyper-parameters on a set of tasks from the literature;
2. It produces a comprehensive measure of evaluation for each model called the *PCE (Principal Component Evaluation)*;
3. It has been implemented as a pip-python package, available to the whole community[2]. Anyone can contribute by improving it by adding (domain-specific) benchmarks and tasks.

[2]The package is publicly available at https://pypi.org/project/vec2best/.

6.2 Methodology

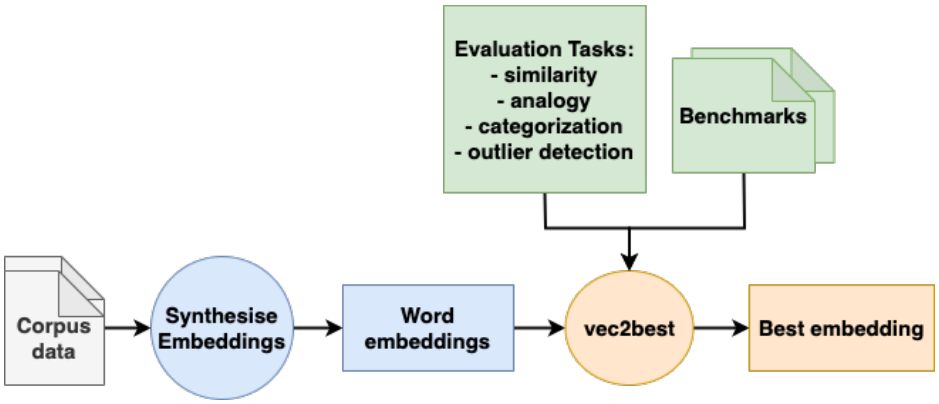


Figure 6.1 The workflow of `vec2best`

As seen in the previous section, there already are a fair number of intrinsic evaluation methods and datasets. *What is missing is a unified method of evaluation that merges the results of those methods to provide a comprehensive measure of the model's worth.* That is the motivation behind `vec2best`: a pipeline to execute the main intrinsic evaluation tasks to provide an extensive and overall evaluation - called *PCE* - of word embeddings.

`vec2best` enables the users to obtain an overall evaluation of their word embeddings, providing at the same time insights regarding the performance achieved by the models on the different tasks and benchmarks that are state-of-the-art methods for intrinsic evaluation.

`vec2best` considers four state-of-the-art tasks, namely: *word similarity*, *word analogy*, *concept categorisation*, and *outlier detection*. The choice to focus on these four tasks follows from what Wang et al. [148] found in their correlation analysis to study the performance consistency of extrinsic and intrinsic evaluators. They argue that word similarity, word analogy, and concept categorisation are the most effective intrinsic evaluators. Since intrinsic evaluators can perform differently for different downstream tasks, they suggest using these three intrinsic evaluators jointly when testing a new word embedding model. Regarding outlier detection Wang et al. [148] say that with larger and more reliable datasets available, they expect the outlier detection task to have better performance, and this is why we included it too.

To evaluate each of these tasks, `vec2best` uses several benchmarks, as shown in Table 6.1 (more details about these benchmarks can be found in Section 6.4.2). The overall evaluation

Table 6.1 The benchmarks considered for each task and their evaluation metric.

Task	Evaluation	Metric	Benchmark	Source
Similarity [74]	Spearman correlation	Cosine similarity	WS353	[59]
			RG65	[128]
			RW	[99]
			MEN	[32]
			MTurk287	[122]
			SimLex999	[78]
			MC30	[109]
			MTurk771	[75]
			YP130	[157]
			Verb143	[15]
			SimVerb3500	[65]
			SemEval17	[35]
			WS353REL	[2]
			WS353SIM	[2]
Analogy [6]	Accuracy	3CosAdd 3CosMul	Google	[106]
		3CosAdd 3CosMul	MSR	[108]
	Spearman correlation	3CosAdd	SemEval2012	[87]
Categorisation [76]	Purity	Clustering	AP	[7]
			BLESS	[19]
			BM (battig)	[20]
			ESSLLI 1a	[18]
			ESSLLI 2b	[18]
			ESSLLI 2c	[18]
Outlier detection [33]	Accuracy	Compactness score	8-8-8	[33]
			WordSim500	[23]

for each task can be computed in three different ways: as the mean, as the minimum, or as the maximum of the evaluation obtained for all its benchmarks. For the analogy task, we need to re-scale the evaluation on SemEval2012 to make it range between $[0, 1]$ as the evaluations on the other tasks, since the evaluation metric is different (Spearman correlation vs. accuracy).

Then for each one of these possible evaluations and each task, we compute the first principal component obtained through the Principal Component Analysis (PCA).

PCA is one of the most widely used methods to reduce the dimensionality of a dataset such that most of the information in the data is preserved. PCA achieves that through the principal components (PCs), which are linear functions of those in the original dataset, uncorrelated with each other while keeping a maximum variance. The problem of finding the PCs reduces to solving an eigenvalue/eigenvector problem [84].

It is common practice to standardise the variables to avoid problems with the covariance matrix. Starting from a $n \times p$ data matrix $\mathbf{X}$, where n is the number of word embedding models evaluated, and p is the number of tasks on which those models were evaluated, each data value x_{ij}—i-th model and j-th task— is centred and divided by the standard deviation s_j of the n evaluations computed for task j: $z_{ij} = (x_{ij} - \bar{x}_j)/s_j$.

Given the standardised variables $\mathbf{z}_1, \ldots, \mathbf{z}_p$ and their correlation matrix $\mathbf{R}$, the eigenvectors $\mathbf{a}_k$ of $\mathbf{R}$ define the $q < p$ uncorrelated maximum-variance linear combinations

$$\mathbf{Z}\mathbf{a}_k = \sum_{j=1}^{p} a_{jk}\mathbf{z}_j, \qquad k \in \{1, \ldots, q\} \tag{6.1}$$

that are called principal components (PCs). The matrix $\mathbf{Z}$ in (6.1) is a $n \times p$ matrix having columns $\mathbf{z}_1, \ldots, \mathbf{z}_p$. By keeping just the first PC - the one that explains most of the variance present in the original p variables - we can obtain a single measure of evaluation:

$$PCE^* = \mathbf{Z}\mathbf{a}_1 = \sum_{j=1}^{p} a_{j1}\mathbf{z}_j. \tag{6.2}$$

To improve the interpretability of such a measure and make it easier to compare the performance of different models, we consider the normalised PCE^* measure:

$$PCE = \frac{PCE^* - \min(PCE^*)}{\max(PCE^*) - \min(PCE^*)} \in [0, 1]. \tag{6.3}$$

Values of *PCE* equal to 0 and 1 are associated with models characterised by the lowest and larger value of PCE^*, respectively. It is possible to show that $\rho_{1j} = \text{Corr}\left(\mathbf{Z}\mathbf{a}_1, \mathbf{z}_j\right) = a_{1,j}\sqrt{\lambda_1}$, where λ_1 is the largest eigenvalue of the matrix $\mathbf{R}$ and $a_{1,j}$ is the j-th element of $\mathbf{a}_1$. Thus, the correlation coefficients ρ_{1j}, $j = 1, \ldots, p$, have concordant signs only if $\mathbf{a}_1$ contains concordant values. Thanks to the Perron-Frobenius theorem [113], this condition is satisfied when the elements in $\mathbf{R}$ are positive, which typically is the case if metrics $\mathbf{z}_1, \ldots, \mathbf{z}_p$ have a similar interpretation (e.g., higher values are associated with better performances). In the case that $\rho_{1j} > 0$ for each j, then higher values of $\mathbf{z}_1, \ldots, \mathbf{z}_p$ lead to higher values of *PCE* and, consequently, $PCE = 1$ is associated to the best model. Differently, if $\rho_{1j} \leq 0$ for each j, them $PCE = 1$ is associated with the worst model.

Since we aim at defining a metric that takes value 1 when it evaluates the best model, in the case $\rho_{1j} \leq 0$, we define

$$PCE = 1 - \frac{PCE^* - \min(PCE^*)}{\max(PCE^*) - \min(PCE^*)}. \tag{6.4}$$

The evaluation of each one of the four tasks - Similarity, Analogy, Categorisation, and Outlier Detection - gives us three different values, namely: PCE^{MIN}, PCE^{MAX}, and PCE^{MEAN}, which refer to the minimum, the maximum, and the mean of the evaluations of each task over the different benchmarks and metrics considered, as in Tab. 6.1. In this way, we have an overall evaluation based on the worst-case scenario (the conservative one), one based on the best scenario, and the last based on a synthesis of all the evaluations for each task. The standard measure of the quality of a given PC is the proportion of total variance that it accounts for:

$$\pi_j = \frac{\lambda_j}{\sum_{j=1}^{p} \lambda_j}, \tag{6.5}$$

where λ_j is the j-th largest eigenvalue of $\mathbf{R}$.

It is well-known that the presence of outliers may impact the resulting PCs since they typically enlarge marginal variances. Nonetheless, in our framework, an outlier refers to a model performing particularly well (or badly) in one or more metrics. Since we aim to detect the best-performing model, outliers are particularly informative.

6.3 vec2best as an Off-the-Shelf Tool

Finally, vec2best has been implemented as a pip-python package, and it can be invoked using a few lines of code shown below:

```
from vec2best.functions import compute_pce
path_to_model = 'data/example_models'
result = compute_pce(path_to_model)
```

The function compute_pce has other six parameters (categorization=True, similarity=True, analogy=True, outlier_detection=True, pce_min=True, pce_max=True, pce_mean=True) set by default as True, and so the output consists in the evaluation of the models over the three tasks and over the PCE^{MIN}, PCE^{MAX}, PCE^{MEAN}. By setting some of those parameters as False, the *PCE* can be computed over a subset of those tasks or the evaluation could be computed only for one or two of the three types of *PCE*.

The output is saved, and the output on the screen shows the percentage of explained variance of the first principal component, and the top 3 models according to the chosen *PCE*. The following is an example of the output:

```
from vec2best.functions import compute_pce
path_to_model = 'data/example_models'
compute_pce(path_to_model, analogy=False, outlier_detection=False,
pce_max=False, pce_mean=False)
```

```
PCE min - percentage of explained variance: 0.95
                                    cat    sim    PCE_min
example_models/ft_0_5_50_5.vec      0.38   0.29   1.00
example_models/glove_5_50_5.vec     0.41   0.25   0.94
example_models/wv2_model_11.vec     0.24   0.17   0.34
```

6.4 Experimental Results

6.4.1 Datasets and Settings for Reproducibility

Training corpus For consistency, we performed the training of the different models on the same corpus, and following [130], we chose the Wikipedia dump (2008-03-01). The used dump already includes the pre-processed version of data that we used as our data without performing further cleaning.

Models For the training, we used three of the most important methods of non-contextual word representations from large text corpora: word2vec [106], fastText [24], and GloVe [117].

The grid of parameters for word2vec and fastText is as follows:

- *algorithm* ∈ {CBOW, SG (skip-gram)}[3]
- *epochs* ∈ {5, 10, 25}
- *size* ∈ {50, 100, 150, 300}
- *window* ∈ {5, 10, 15}

For GloVe, we considered the same hyper-parameters except for the *algorithm*. That accounts for 135 word embedding models trained in total. We chose the fastText parameters following the suggestions on the fastText site [4]. The fastText and word2vec models were trained using the Gensim library[5], while the GloVe models were trained using the officially released toolkit[6].

The part of the evaluation on the tasks of similarity, analogy, and categorisation is based on the Python package word-embeddings-benchmarks [7] [81]. The part of the evaluation on the task of outlier detection is based on the code and benchmarks provided by Camacho-Collados and Navigli [33] [8] and by Blair et al. [23] [9].

[3] only for fastText and word2vec

[4] https://fasttext.cc/docs/en/unsupervised-tutorial.html

[5] https://radimrehurek.com/gensim/index.html

[6] https://nlp.stanford.edu/projects/glove/

[7] https://github.com/kudkudak/word-embeddings-benchmarks

[8] http://lcl.uniroma1.it/outlier-detection/

[9] https://github.com/peblair/wiki-sem-500

6.4.2 Benchmarks

The benchmarks used are listed in Table 6.1.

Word Similarity

We chose 14 datasets for word similarity evaluation. Among these datasets, WS353, WS353SIM, WS353REL, and RW are the more popular because of their high-quality word pairs. The characteristics of these benchmarks are the following:

- WS353 has 353 pairs assessed by semantic similarity with a scale from 0 to 10.
- RG65 has 65 pairs assessed by semantic similarity with a scale from 0 to 4.
- RW has 2 034 pairs of words with low frequency (rare words) assessed by semantic similarity with a scale from 0 to 10.
- MEN has 3 000 pairs assessed by semantic relatedness with a discrete scale from 0 to 50.
- MTurk287 has 287 pairs assessed by semantic relatedness with a scale from 0 to 5.
- SimLex999 has 999 pairs assessed by semantic similarity with a scale from 0 to 10.
- MC30 has 30 pairs, a subset of RG65 which contains 10 pairs with high similarity, 10 with middle similarity, and 10 with low similarity.
- MTurk771 has 771 pairs assessed by semantic relatedness with a scale from 0 to 5.
- YP130 has 130 pairs of verbs assessed by semantic similarity with a scale from 0 to 4.
- Verb143 has 143 pairs of verbs assessed by semantic similarity with a scale from 0 to 4.
- SimVerb3500 has 3 500 pairs of verbs assessed by semantic similarity with a scale from 0 to 4.
- SemEval17 has 500 pairs assessed by semantic similarity with a scale from 0 to 4 prepared for the SemEval-2017 Task 2.
- WS353REL has 252 pairs, a subset of WS353 containing no pairs of similar concepts.
- WS353SIM has 203 pairs, a subset of WS353 containing similar or unassociated (to mark all pairs that receive a low rating as unassociated) pairs.

Word Analogy

For word analogy evaluation, the Google dataset contains 19 544 questions, divided into *semantic* and *morpho-syntactic* categories, each with 8 869 and 10 675 questions. The MSR dataset contains 8 000 analogy questions divided into 16 morphological classes. Both 3CosAdd and 3CosMul inference methods are implemented. Lastly, SemEval2012 contains 10 014 questions divided into 10 semantic classes and 79 sub-classes.

Word Categorisation

Four datasets are used in concept categorisation evaluation. The AP dataset contains 402 words that are divided into 21 categories. The BLESS dataset consists of 200 words divided into 27 semantic classes. The BM dataset is the larger, with 5 321 words divided into 56 categories. Lastly, ESSLLI 1a contains 44 words divided into 6 semantic classes, ESSLLI 2b contains 40 words divided into 3 semantic classes, and ESSLLI 2c contains 45 words divided into 9 semantic classes.

Outlier Detection

For the outlier detection task, the 8-8-8 dataset has 8 clusters, each one represented by a set of 8 words with 8 outliers. The WordSim500 consists of 500 clusters, each one represented by a set of 8 words with 5 to 7 outliers.

6.4.3 Results

In Tables 6.2, 6.3, 6.4, 6.5 we show the results obtained by the 135 word embedding models (see Sec. 6.4.1) - grouped by the five different types of models (GloVe, fastText CBOW, and SG, word2vec CBOW and SG) - over the four tasks (similarity, analogy, categorisation, and outlier detection) and their benchmarks.

The models with the best results overall are the word2vec (CBOW and SG), which achieve - on average - the best results in 13 out of 14 benchmarks for similarity (only for MC30 the best result is obtained only by the fastText CBOW models), and 4 out of 6 for categorisation. In the analogy task, the word2vec CBOW models achieve, on average, the

best results on 3 tasks out of 5. For the task of outlier detection, word2vec CBOW and SG obtain the same results, which are, on average, the best over both benchmarks.

FastText CBOW over-performs fastText SG on most benchmarks, and these models achieve the best performance on average over 10 benchmarks out of 27 in total (versus only 1 time over 27 for fastText SG.

GloVe models achieve the best performance on average only for the categorisation task (on BLESS and ESSLLI 1a benchmarks).

Overall, we can say that word2vec SG is, on average, the best performing on the task of similarity because those word embeddings achieve the best results on 10 out of 14 of the benchmarks (see Table 6.2). FastText and word2vec CBOW models obtain the best results on average on 3 benchmarks out of 5 for analogy. word2vec SG models are the best performing for the categorisation task (3 out of 6 benchmarks), and, lastly, for outlier detection, word2vec CBOW and SG models obtain, on average, the best results over both benchmarks considered.

Table 6.2 Mean and standard deviation obtained by the models - grouped by the five different types of models - over all the benchmarks for the similarity task.

	WS353	**RG65**	**RW**	**MEN**	**MTurk287**
GloVe	0.44±0.21	0.61±0.26	0.24±0.13	0.55±0.27	0.51±0.24
ft - CBOW	0.59±0.03	0.76±0.02	0.38±0.02	**0.73±0.02**	**0.67±0.01**
ft - SG	0.56±0.04	0.77±0.02	**0.39±0.03**	0.68±0.03	0.63±0.02
w2v - CBOW	0.6±0.02	**0.79±0.01**	0.37±0.02	0.72±0.02	**0.67±0.01**
w2v - SG	**0.66±0.03**	0.77±0.03	**0.39±0.02**	**0.73±0.03**	**0.67±0.01**
	SimLex999	**MC30**	**MTurk771**	**YP130**	**Verb143**
GloVe	0.23±0.13	0.59±0.25	0.5±0.25	0.38±0.19	0.37±0.19
ft - CBOW	**0.35±0.04**	**0.76±0.02**	0.62±0.03	0.39±0.05	0.39±0.05
ft - SG	0.29±0.03	0.71±0.04	0.58±0.03	0.44±0.03	0.45±0.03
w2v - CBOW	**0.35±0.04**	0.74±0.02	**0.63±0.02**	0.4±0.05	0.4±0.04
w2v - SG	0.33±0.04	0.75±0.03	**0.63±0.03**	**0.46±0.05**	**0.49±0.05**
	SimVerb 3500	**SemEval17**	**WS353REL**	**WS353SIM**	
GloVe	0.12±0.07	0.36±0.14	0.38±0.18	0.55±0.27	
ft - CBOW	0.22±0.03	**0.51±0.02**	0.51±0.03	0.68±0.02	
ft - SG	0.17±0.02	0.48±0.02	0.49±0.05	0.65±0.04	
w2v - CBOW	**0.23±0.02**	0.49±0.01	0.5±0.02	0.72±0.01	
w2v - SG	0.2±0.03	**0.51±0.01**	**0.61±0.03**	**0.73±0.02**	

Table 6.3 Mean and standard deviation obtained by the models - grouped by the five different types of models - over all the benchmarks for the analogy task.

	Google Add	**Google Mul**	**MSR Add**	**MSR Mul**	**SemEval 2012**
GloVe	0.53±0.1	0.47±0.14	0.38±0.07	0.34±0.09	0.15±0.02
ft - CBOW	0.45±0.05	0.43±0.1	**0.5±0.06**	**0.46±0.12**	**0.16±0.02**
ft - SG	0.51±0.09	0.49±0.12	0.42±0.11	0.41±0.13	0.14±0.02
w2v - CBOW	**0.61±0.09**	**0.55±0.13**	0.44±0.09	0.38±0.14	**0.16±0.02**
w2v - SG	0.54±0.11	0.51±0.14	0.36±0.11	0.33±0.14	0.15±0.02

Table 6.4 Mean and standard deviation obtained by the models - grouped by the five different types of models - over the benchmarks for categorisation.

	AP	**BLESS**	**BM**
GloVe	0.62±0.02	**0.78±0.03**	0.41±0.01
ft - CBOW	0.6±0.02	0.69±0.04	0.39±0.01
ft - SG	0.62±0.03	0.73±0.06	0.43±0.01
w2v - CBOW	0.63±0.02	0.73±0.04	**0.44±0.01**
w2v - SG	**0.64±0.03**	0.75±0.06	**0.44±0.01**
	ESSLLI 2c	**ESSLLI 2b**	**ESSLLI 1a**
GloVe	0.6±0.02	0.75±0.02	**0.78±0.03**
ft - CBOW	**0.62±0.03**	0.74±0.01	0.76±0.03
ft - SG	0.6±0.04	0.74±0.04	0.73±0.05
w2v - CBOW	0.6±0.02	**0.78±0.03**	0.77±0.02
w2v - SG	**0.62±0.04**	0.77±0.04	0.77±0.04

We can consider the results of the word embedding evaluations robust if the best model does not change over different ways of computing the PCE. To this end, we consider three different PCE evaluations (PCE^{MIN}, PCE^{MAX}, PCE^{MEAN}) and a combination of two of those, computed as follows:

$$PCE_i = \alpha \cdot PCE_i^{MIN} + (1-\alpha) \cdot PCE_i^{MAX}, \tag{6.6}$$

where $\alpha \in [0,1]$ and $i = 1,\ldots,n$. Then we check which is the best model in each one of these combinations and according to PCE^{MEAN} too.

Table 6.5 Mean and standard deviation obtained by the models - grouped by the five different types of models - over the benchmarks for outlier detection.

	8-8-8	**WordSim500**
GloVe	0.13±0.04	0.46±0.15
ft - CBOW	**0.16±0.01**	0.51±0.01
ft - SG	0.12±0.03	0.53±0.01
w2v - CBOW	**0.16±0.02**	**0.57±0.01**
w2v - SG	**0.16±0.01**	**0.57±0.01**

In Tab. 6.6, we can see that in 9 cases out of 12, the best model is word2vec CBOW with 25 epochs, a dimension equal to 300, and a window of size equal to 5. For PCE^{MEAN}, the best model differs from that one only because it is SG instead of CBOW. Only for PCE^{MIN} - the most conservative scenario evaluation - and for the combination with $\alpha = 0.1$, the best model is word2vec SG with 25 epochs, a dimension equal to 300 and a window of size of 10. We can say that this analysis is a strong hint at the robustness of `vec2best` since the best model is very consistent with different final evaluations, with only two parameters diverse in 3 cases out of 12.

Figure 6.2 shows that the correlation of the overall evaluation for the three tasks performed by `vec2best` with the final evaluation metric *PCE* is high, ranging from 0.38 to 0.91, with the only exception of a modest correlation between the PCE^{MIN} and the final evaluation for the categorisation task (0.09). This result was achieved thanks to the PCA, which reduces the dimensionality while preserving most of the information in the data.

The four tasks - analogy, similarity, categorisation, and outlier detection - show a less strong correlation with one another, except similarity and outlier detection, which display a correlation of 0.9. The task that has the lowest correlation with the others is the categorisation one. That leads us to think that by using these four tasks for computing the *PCE*, we are not considering redundant information, and each task contributes valuable information to the final evaluation.

In Figure 6.3, we can see the three scatter-plots that show the relation between PCE^{MAX}, PCE^{MEAN} and PCE^{MIN} shown in couples. We can see that the three measures display a linear joint distribution and also that all three report the majority of models with a high *PCE* value, while there are few models with a low *PCE*, separated from the others.

Table 6.6 The evaluation of the best and worst model according to different configurations of PCE.

	Model	**PCE**
α=0.0 (PCE^{MIN})	w2v_vectors_SG_25_300_10	1.00
α=0.1	w2v_vectors_SG_25_300_10	0.99
α=0.2	w2v_vectors_CBOW_25_300_5	0.98
α=0.3	w2v_vectors_CBOW_25_300_5	0.98
α=0.4	w2v_vectors_CBOW_25_300_5	0.98
α=0.5	w2v_vectors_CBOW_25_300_5	0.98
α=0.6	w2v_vectors_CBOW_25_300_5	0.99
α=0.7	w2v_vectors_CBOW_25_300_5	0.99
α=0.8	w2v_vectors_CBOW_25_300_5	0.99
α=0.9	w2v_vectors_CBOW_25_300_5	1.00
α=1.0 (PCE^{MAX})	w2v_vectors_CBOW_25_300_5	1.00
PCE^{MEAN}	w2v_vectors_SG_25_300_5	1.00

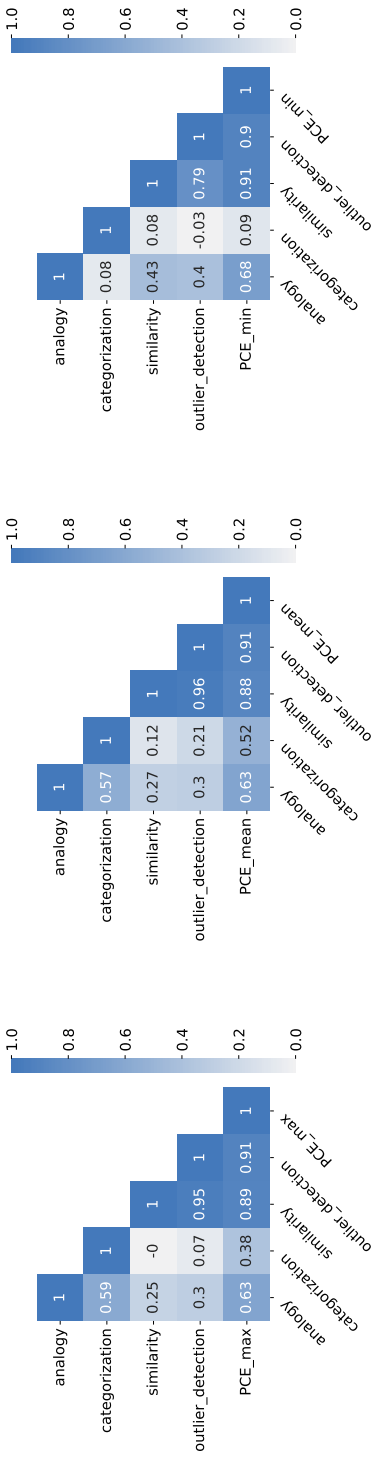


Figure 6.2 The correlation between PCE^{MAX}, PCE^{MEAN} and PCE^{MIN} respectively and the evaluation over the tasks performed by `vec2best`.

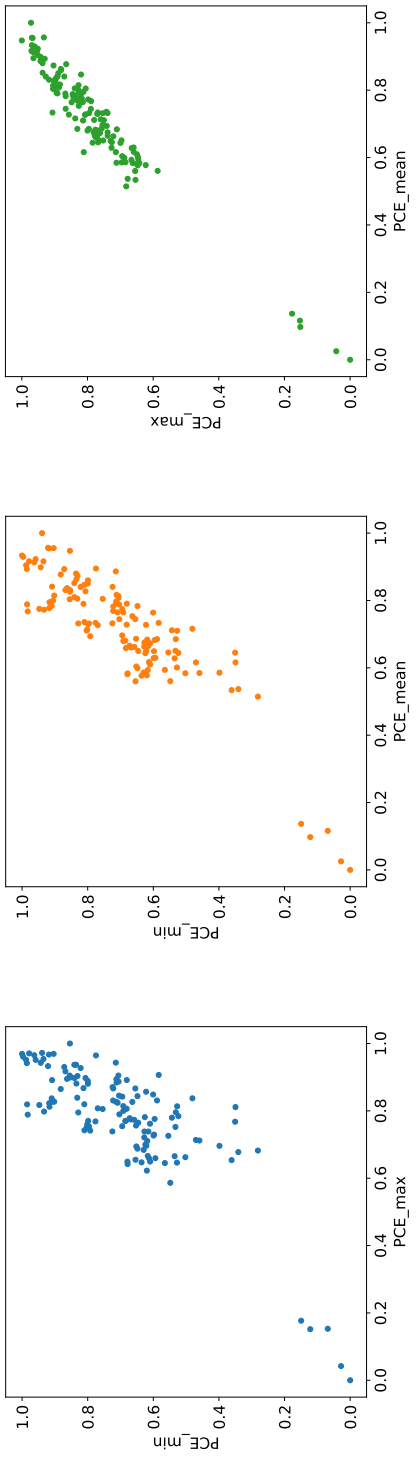


Figure 6.3 The scatter-plots representing the joint distribution between PCE^{MAX}, PCE^{MEAN} and PCE^{MIN}.

In Figures 6.4, 6.5, 6.6, we can see the relation between the hyper-parameters and the three measures of *PCE*, showing that better results are obtained with a bigger vector size for each type of model; fastText SG models and GloVe were the only ones to get better results with smaller window size. Except for the GloVe models, the number of epochs is the parameter that seems to make the smallest difference in the final performance of the word embedding model.

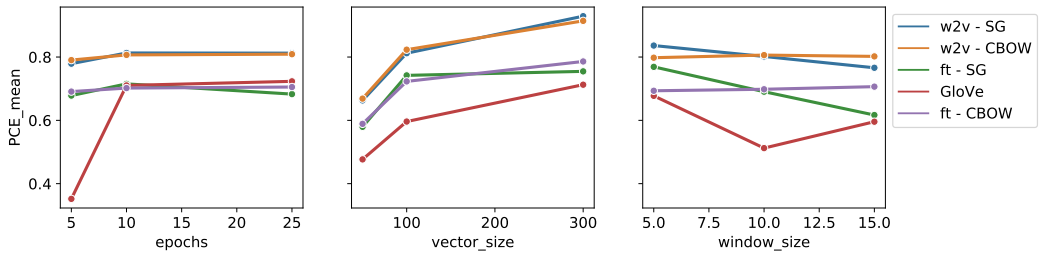


Figure 6.4 The performance of the models according to PCE^{MEAN} with different hyper-parameters configurations.

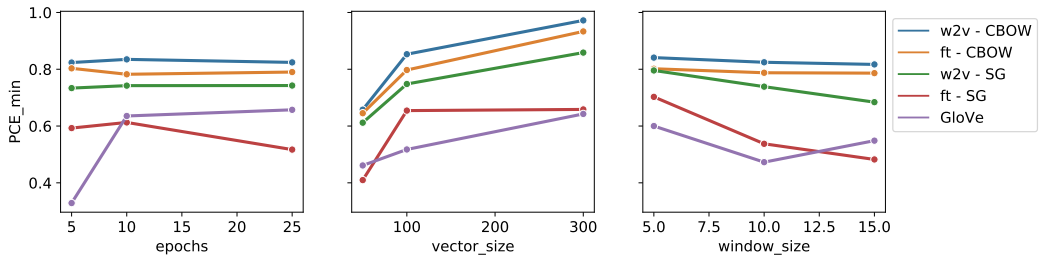


Figure 6.5 The performance of the models according to PCE^{MIN} with different hyper-parameters configurations.

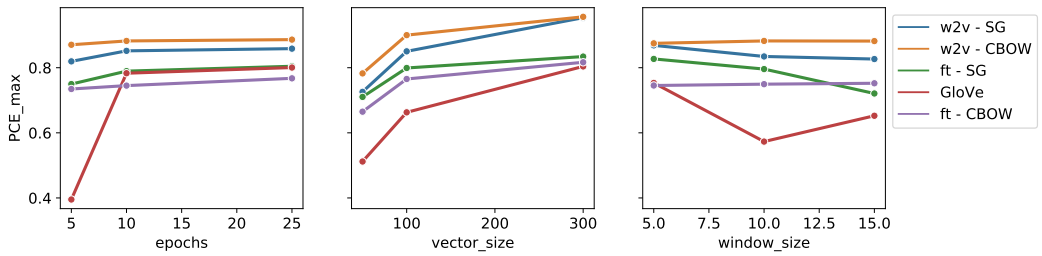


Figure 6.6 The performance of the models according to PCE^{MAX} with different hyper-parameters configurations.

Part III

Taxonomy Enrichment and Alignment via Word Embeddings

7

Setting the Stage for Taxonomy Enrichment and Alignment

In Chapter 3, we defined what a taxonomy is and how to measure semantic similarity between words in a taxonomy. In this Chapter, we will introduce the topic of taxonomy induction from free text. Secondly, we will expose different methodologies to enrich a taxonomy with new and pertinent concepts. Lastly, we will approach how the alignment of different taxonomies concerning the same field has been dealt with in the literature.

7.1 Taxonomy Induction

With massive Web data available, several taxonomies are constructed from human-compiled resources such as Wikipedia or Wikidata, see for example [119, 103, 133]. Several methods have been developed to induce taxonomies from text corpora even though this task is far from being solved, given that inducing taxonomies from text corpora has three main drawbacks [149]:

- Text corpora may vary in size, topic and quality, which makes it unlikely to develop a single solution;
- The accuracy of free-text taxonomies is usually lower than the accuracy of many Wikipedia-based ones;
- The task is still insufficiently studied in emerging and specific domains and for non-English languages.

At the current state, a free-text-based taxonomy construction system typically operates in three steps [149]: (i) extracting *is-a* relations, (ii) constructing a complete taxonomy from *is-a* relations, and (iii) cleansing the taxonomy created (i.e. removing relations that create cycles).

We will explore these three steps in the following sections.

7.1.1 Extracting *is-a* Relations

The most used methods for extracting *is-a* relations can be divided into pattern-based methods and distributional approaches.

Pattern-based Methods

Pattern-based methods extract the *is-a* relation between the terms x and y based on the lexico-syntactic paths connecting x and y in a corpus, i.e. if x and y appear in the same sentence and satisfy a particular pattern.

A typical pattern is "[C] such as [E]", where [C] and [E] are placeholders of noun phrases that are regarded as the hypernym y and the hyponym x respectively for an *is-a* relation (x, y) [77].

Despite the successful applications, these patterns are too specific to cover all linguistic circumstances. Simple pattern matching is also prone to error due to idiomatic expressions, parsing errors, incomplete extractions and ambiguous issues. Moreover, using patterns as features may result in the sparsity of the feature space, and these methods are overly language-dependent.

Distributional Approaches

Distributional approaches predict *is-a* relations between terms based on their distributional representations by either unsupervised or supervised models.

Unsupervised measures Early works of distributional similarity measures mostly focus on symmetric measures such as cosine similarity or Jaccard; asymmetric measures model the asymmetric property of *is-a* relations, following the Distributional Inclusion Hypothesis (DIH) [64], that assumes that a hyponym only appears in some of its hypernym's contexts, while a hypernym appears in all contexts of its hyponyms.

Supervised measures With training sets available, classification and ranking methods train a model to predict hypernymy based on the representations of a term pair (x, y). The most popular representations for x and y are word embeddings generated by pre-trained neural language models such as word2vec or GloVe (see Chapter 2).

Hypernym generation approaches directly model how to generate hypernyms based on the representations of hyponyms in the embedding space: they predict a pair (x, y) by whether the model can map $\vec{x}$ to a vector close to $\vec{y}$.

Their main disadvantages are that they are less precise in detecting specific *is-a* relations and tend to discover broad semantic similarities, hypernymy relations can sometimes confused with synonymy or co-hyponymy. Moreover, the models are heavily dependent on the training set.

7.1.2 Taxonomy Induction

There are three main approaches for taxonomy induction from *is-a* relations, which are presented in the following sub-sections.

Incremental Learning

Several methods construct an entire taxonomy from a *seed* taxonomy via incremental learning. The extracted terms can either refer to existing entities in the taxonomy or new ones and, for example, Shen et al. [135] propose a graph-based method to link these terms to the

taxonomy or insert new entities into the taxonomy. These methods are similar to the taxonomy enrichment methods that we will discuss in the following section.

Clustering

Taxonomy learning can be modelled also as a *clustering* problem, where similar terms clustered together may share the same hypernym. Alfarone and Davis [5] propose a method in which the lowest common ancestor of a collection of terms clustered by K-Medoids is inferred as their common hypernym.

Graph-based Approaches

Graph-based approaches are naturally suitable for this task because taxonomies can generally be represented as graphs. A frequently used algorithm is the optimal branching algorithm [146]: it first assigns edge weights based on graph connectivity (e.g., in-degree, betweenness) and finds an optimal branching based on Chu-Liu/Edmonds's algorithm [89]. After noisy edge removal, a rooted tree is constructed with maximum weights.

7.1.3 Taxonomy Cleansing

The final step of taxonomy learning is taxonomy cleansing, which removes wrong *is-a* relations to improve the quality.

Incorrect *is-a* relations may arise in taxonomies in the form of cycles, or with the issue of entity ambiguity.

The transitivity property does not necessarily hold in automatically constructed taxonomies, for example, the facts *(Albert Einstein, is-a, professor)* and *(professor, is-a, position)* do not mean that *(Albert Einstein, is-a, position)* [149].

7.2 Taxonomy Enrichment

In the past literature, despite the automatic construction of taxonomies from scratch having received considerable attention [149], the same cannot be said for the augmentation of existing hierarchies. Most of the work in the area of automated taxonomy enrichment relies

heavily on domain-specific knowledge [21, 58] or lexical structures specific to an existing resource, like the WordNet synset [138, 88, 129] or Wikipedia categories [137]. In recent years, some scholars have tried to overcome those limitations by developing methodologies for the automated enrichment of generic taxonomies.

Wang et al. [150] use a hierarchical Dirichlet model to construct a complete taxonomy for an online corpus and associate each document in the corpus with the relevant categories from this complete taxonomy. The framework has three steps: (i) detecting meaningful missing categories, (ii) modelling the category hierarchy using a hierarchical Dirichlet model and predicting the optimal tree structure according to the model, and (iii) reorganising the corpus using the complete category structure.

The missing category discovery is achieved with a semi-supervised learning method, using a combination of labelled data and search click log data to augment the labelled one. To generate a classifier the authors use Linear Support Vector Machine (SVM). For the step of hierarchy reconstruction, they use a hierarchical Dirichlet model to capture the generating process of a taxonomy based on the content of the item pages and search click log data from users. The authors formulate the problem of finding the optimal tree as a structure learning problem solved by the Maximum Spanning Tree (MST) algorithm for directed graphs. The last step is item page re-tagging, aimed at augmenting the category set for each item page with relevant new categories: given a set of potential new categories they use a multi-label classification method based on a set of features, including centroid-based similarity features, click features and features derived from relationships among categories.

Vedula et al. [145] use word embeddings to find semantically similar concepts in the taxonomy. They create word embeddings from a text corpus to represent new concepts and taxonomy concepts, and thanks to those, they find for each new concept its k-Nearest Neighbours (k-NN), limiting the research for the new concept's parents to their ancestors. Then, using a combination of graph-theoretic features and semantic similarity-based features, they predict the potential parent-child relationship between existing concepts and new concepts. The graph-theoretic features they use are:

- Katz similarity: for each new word x and potential parent p they compute:

$$KS(x,p) = \sum_{l=1}^{l_{MAX}} \eta^l |paths_l(x,p)| \tag{7.1}$$

- Random walk betweenness centrality: they consider all combinations of pairs of x's k-NN and compute random walks between them. Then, they compute the betweenness centrality for each p.
- Information propagation score: it aims at propagating weight from x's k-NN following relations connecting hyponym and hypernym:

$$IP(v) = \begin{cases} w_0(v) & v \in N_x \\ \sum_{(u,v \in E)} \frac{(1-\delta)IP(u)}{p(u)^{\alpha} e^{ld(v)\beta}} & v \notin N_x, u, v \in V \end{cases} \tag{7.2}$$

On the other hand, the semantic similarity-based features are:

- Ancestor-neighbour similarity: for each x they compute the pairwise term overlap between the potential parent's text and each k-NN, merging the results using the mean of the overlaps.
- New-concept similarity: they compute Jaccard similarity between words in the text of the new term with those in the text of the potential parent.
- Pointwise mutual information:

$$PMI(x, p) = \log\left(\frac{num(x, p)}{num(x)num(p)}\right) \tag{7.3}$$

with *num* number of occurrences or co-occurrences of terms.

This work by Vedula et al. [145] was evaluated on Wikipedia and WordNet.

Aly et al. [8] use the similarity between Poincaré term embeddings to find *orphans* (disconnected nodes) and *outliers* (child assigned to wrong parents) in a taxonomy. They create Poincaré embeddings using hypernym relationships extracted from a combination of general (e.g. Wikipedia) and domain-specific (web pages) corpora using lexical-syntactic patterns. These Poincaré embeddings use a hyperbolic space and are designed for modelling hierarchical relationships between words as they explicitly capture the hierarchy between them in the embedding space. These embeddings are used to compute a rank between every child x and parent y of the existing taxonomy, defined as the index of y in the list of sorted Poincaré distances of all entities of the taxonomy to x: these ranks are used to place new concepts (orphans) and misplaced concepts (outliers) in the taxonomy.

Shen et al. [134] use a set of <query concept, anchor concept> from an existing hierarchy to train a model to predict the parent-child relation between the anchor and the query concepts: the whole procedure is called TaxoExpan.

Given an existing taxonomy $\mathcal{T}^0 = (\mathcal{N}^0, \mathcal{E}^0)$, TaxoExpan generates a set of <query concept, anchor concept> using as positive examples a child (n_c) and a parent (n_p) that are in a relation n_c *is-a* n_p, while as negative examples n_c and N randomly sampled nodes. They model the matching score between a query concept n_i and an anchor concept a_i by projecting them into a vector space, using a position-enhanced graph neural network, whose key idea is to learn a set of position embeddings and enrich each node feature with its corresponding position embedding. Using such self-supervision data, TaxoExpan learns a model to predict whether a query concept is the direct hyponym of an anchor concept. For each new concept, they run the inference procedure and find its best parent node in the existing taxonomy, solving:

$$a_i^* = \arg\max_{a_i \in \mathcal{N}^0} \log P(n_i | a_i, \Theta) \tag{7.4}$$

It was evaluated on Field-of- Study (FoS) Taxonomy in Microsoft Academic Graph (MAG).

Table 7.1 The related work on taxonomy enrichment.

Paper	Methodologies used	Datasets for evaluation
[150]	Hierarchical Dirichlet model	Datasets from a commercial search engine
[145]	Word embeddings, k-NN	Wikipedia, WordNet
[8]	Poincaré word embeddings	Data of SemEval2016 TExEval
[134]	Graph neural network	Microsoft Academic Graph (MAG)

In Chapter 8, we present `NEE`, a framework that leverages word embeddings to estimate the degree to which data conforms to a given taxonomy, identifying new entities and concepts for the taxonomy.

7.3 Taxonomy Alignment

In recent literature, taxonomy alignment has received considerable attention, and different approaches have been proposed to create an efficient mapping.

In one of the first approaches, Euzenat et al. [54] compute the similarity between entities through a system of quasi-linear equations, which start from lexical similarity derived by WordNet 2.0 and gradually includes contributions from structure comparing functions.

Avesani et al. [13] use a syntactic and a semantic score of taxonomic similarity, called COMA and S-match, respectively. COMA exploits element and structure-level syntactic similarity, while S-match uses WordNet 2.0 to derive semantic similarity between words.

Wu et al. [154] propose an approach based on Wikipedia-matching, and keywords are considered to perform document classification without employing standard occurrence methods.

Despite being relevant and widely used, those methods are built on specific lexical resources (WordNet, Wikipedia) and thus are not suitable for different domains.

Jung [86] uses LSA (Latent Semantic Analysis) to group sets of related entities based on their co-occurrence matrix and TF-IDF, also considering the description of the taxonomic concepts.

Wu et al. [151] train a bilingual topic model on contextual text extracted from the web to build semantic vectors of the topics of two multi-lingual taxonomies. The cosine similarity between those vectors represents the relevance of each concept in the source taxonomy with its candidate-matched categories. Then, each candidate entity is evaluated through syntactic similarity.

Da Silva et al. [48] present Alin, an interactive ontology matching approach that uses expert feedback to approve or reject selected mappings and to dynamically improve the set of mappings through the use of anti-patterns. Alin chooses the first mappings through semantic and lexical similarity metrics, then suspends some using semantic criteria. It then uses expert feedback and mapping anti-patterns to reject the inconsistent mappings with those accepted by the expert.

Real et al. [123] developed an approach in which matches can use domain-specific lexicon and grammar to improve their performance when matching domain-knowledge resources.

Lv and Peng [100] propose a periodic learning optimisation model based on an interactive grasshopper optimisation algorithm is proposed. They also introduce a roulette wheel approach to select the most problematic mappings and reward or punishment mechanisms to propagate user feedback to the evolving population.

Those kinds of approaches make use of contextual information and learning algorithms. However, none of them considers the vertical structure of the taxonomy to match entities nor employ distributional semantics, which has shown to be beneficial in several NLP applications.

Table 7.2 The related work on taxonomy alignment.

Paper	**Methodologies used**	**Datasets for evaluation**
[54]	Similarity-based paradigm	BibTeX/MIT/UMBC, Karlsruhe, INRIA
[13]	Similarity-based paradigm	Google, Looksmart and Yahoo
[154]	Document classification	WordNet, Wikipedia
[86]	Latent Semantic Analysis	ACM Classification, On-line Medical Dictionary (OMD)
[151]	Bilingual Biterm Topic Model (BiBTM)	JD.com/Bay.com, Dmoz.org/Yahoo
[48]	Interactive Ontology Matching	OAEI 2018
[100]	Grasshopper optimisation algorithm	OAEI 2020

In Chapter 9, we propose `WETA`, a domain-independent, knowledge-poor method for automatic taxonomy alignment via word embeddings.

8

Taxonomy Enrichment with Word Embeddings

In Chapter 7, we introduced the topic of taxonomy induction and different methodologies to enrich a taxonomy with new and pertinent concepts. In this Chapter, we present `NEE`, a framework that leverages word embeddings to estimate the degree to which data conforms to a given taxonomy, identifying new entities and concepts for the taxonomy itself[1].

8.1 Setting the Stage

Unlike the automated construction of new taxonomies from scratch, which is a well-established research area [149], the augmentation of existing hierarchies is gaining in importance, given its relevance in many practical scenarios (see Section 7.2 in Chapter 7). Human languages are evolving, and online content is constantly growing. As a consequence, people often need to enrich existing taxonomies with new concepts and items without repeating

[1] Some results of this Chapter were published in [69].

their whole construction process every time. To date, the most adopted approach to enrich or extend standard *de-jure* taxonomies leans on expert panels and surveys that identify and validate which term has to be added to a taxonomy. The approach relies on human knowledge, with no support from the AI side, and this makes the process costly and time-consuming. To extract semantic information from text corpora, we resort to distributional semantics, in which words are represented by semantic vectors, usually derived from a large corpus using co-occurrence statistics or neural network training, and their use improves learning algorithms in many NLP tasks.

Given an existing taxonomy $\mathcal{T}$ - see Def. 3.1.1 in Chapter 3 - the goal of `NEE` is to extend $\mathcal{T}$ with new *mentions* (entities) coming from a text corpus. Each mention is assigned to one or multiple candidate destination nodes of $\mathcal{T}$, along with a score value and a set of measures. More formally:

Definition 8.1.1 (Taxonomy Enrichment Problem (TEP)) *Let $\mathcal{T}$ be a taxonomy as in Def.3.1.1, and let $\mathcal{D}$ be a corpus; a Taxonomy Enrichment Problem (TEP) is a 3-tuple $(\mathcal{M}, \mathcal{H}^m, \mathcal{S})$, where:*

- *$\mathcal{M}$ is a set of mentions extracted from $\mathcal{D}$, i.e., $m \in \mathcal{M}$;*
- *$\mathcal{S} : \mathcal{W} \times \mathcal{M} \rightarrow [0,1]$ is a scoring function that estimates the relevance of m with respect to an existing word w. Ideally, the scoring function might consider the frequency of m, as well as the similarity between m and w according to $\mathcal{D}$.*
- *$\mathcal{H}^m \subseteq \{(c,m) | c \in \mathcal{C} \wedge m \in \mathcal{M}\}$ is a taxonomic relation (edge) existing between a <concept,mention> pair. Intuitively, $\mathcal{H}^m$ models the pertinence of m to be an entity of the existing concept c;*

A solution to TEP computed over $\mathcal{D}$ is a 7-tuple $\mathcal{T}^{\mathcal{D}} = (\mathcal{C}, \mathcal{W}, \mathcal{H}^c, \mathcal{F}, \mathcal{M}, \mathcal{H}^m, \mathcal{S})$.

8.2 Methodology

In this section, we describe our overall approach to enriching a taxonomy with new emerging entities, which is mainly composed of three steps:

- Learn word embeddings;
- Suggest new entities;
- Vote and enrich.

`NEE`'s workflow is shown in Fig. 8.1.

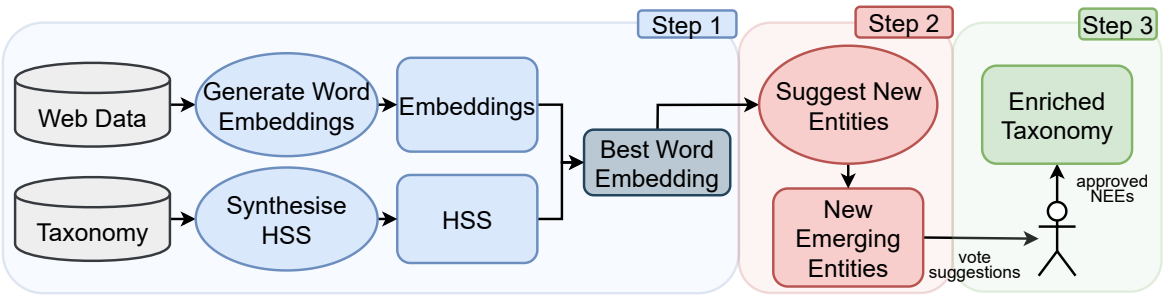


Figure 8.1 A representation of the NEE workflow highlighting the main modules.

8.2.1 Step 1: Synthesise Word Embeddings

The first step requires learning a vector representation of the words in the corpus to preserve the semantic relationships expressed by the taxonomy itself. To accomplish that, we apply the procedure described in Chapter 5. We rely on three distinct sub-tasks, which are the following:

- Train different word embedding models;
- Construct a measure of pairwise semantic similarity between taxonomic elements, namely hierarchical semantic relatedness (HSS);
- Evaluate the embeddings generated in *i)* in terms of correlation between the HSS and the cosine similarity between a pair of terms in the taxonomy.

Step 1.1 Generation of Embeddings We compute the word embeddings associated with all the words in the taxonomy using different models and different hyper-parameters. NEE employs and evaluates three of the most important methods to induce static word embeddings from large text corpora. One, GloVe [117], is based on co-occurrence matrix factorisation while the other two, word2vec [106], and fastText [24], on neural networks training. Notice that FastText considers sub-word information, and this allows one to share information between words to represent rare words, typos and words with the same root.

Step 1.2 Computation of the Hierarchical Semantic Similarity (HSS) We compute the HSS between all the pairs of words in the taxonomy. We defined the HSS in Section 5.2.1 in Chapter 5 as:

$$sim_{\texttt{HSS}}(w_1, w_2) = \sum_{\ell \in \mathcal{L}} \hat{p}\left(\ell = LCA \mid w_1, w_2\right) \times IC(LCA) \tag{8.1}$$

between two words w_1 and w_2.

Step 1.3 Word Embedding Selection Finally, the performance of each word vector model generated in Step 1.1 is assessed - following the `TaxoVec` framework (see 5.2.3) - by the Pearson Correlation of the `HSS` between all the pairs of words in the taxonomy with the cosine similarity between their vectors in the model space. The Correlation coefficient can be interpreted as a measure of the fidelity of the vector model to the taxonomy.

8.2.2 Step 2: Suggest New Entities

Step 2 is aimed at extracting new terms from the corpus and suggesting the most suitable concepts under which they could be positioned in $\mathcal{T}$. To do this, `NEE` works in three distinct steps shown in PseudoCode 1: first, it computes the word embedding models and finds the most appropriate; second, it extracts a set of mentions $\mathcal{M}$ from the corpus $\mathcal{D}$; then, it proposes a set of measures, namely GAS (Generality, Adequacy, and Specificity) to estimate the suitability of a mention $m \in \mathcal{M}$ as an entity of the concepts in $\mathcal{C}$.

PseudoCode 1 `NEE`

Require: $\mathcal{T}(\mathcal{C}, \mathcal{W}, \mathcal{H}^c, \mathcal{F})$ as in Def. 3.1.1; $\mathcal{D}$ dataset;

1: $\mathcal{E} \leftarrow$ best_embedding$(\mathcal{D}, \mathcal{T})$ } *Step*1
2: $\mathcal{M} \leftarrow \varnothing$ //init the set of mentions
3: **for all** $w \in \mathcal{W}$ **do**
4: $\quad \mathcal{M} \leftarrow \mathcal{M} \cup most\_similar(\overrightarrow{\mathcal{E}[w]}, \mathcal{S})$ //ordered according to $\mathcal{S}$ of Eq.8.2
5: **for all** $m \in \mathcal{M}$ **do** } *Step*2
6: $\quad \mathbf{G}_m \leftarrow$ compute Eq.8.3
7: $\quad$ **for all** $c \in \mathcal{C}$ **do**
8: $\quad\quad \mathbf{S}_{m,c}, \mathbf{A}_{m,c} \leftarrow$ compute Eq.8.4, 8.5
9: $\quad\quad \mathcal{H}^m \leftarrow$ user_eval$(m, c, \mathbf{G}_m, \mathbf{A}_{m,c}, \mathbf{S}_{m,c})$ } *Step*3
10: **return** $(\mathcal{M}, \mathcal{H}^m)$

Step 2.1 Extract New Mentions from the Corpus First, we select a starting word w_0 from the taxonomy. Then we consider the top-5 mentions in $\mathcal{D}$ with associated the highest

score value $\mathcal{S}$ with w_0, where the score $\mathcal{S}(m,w)$ of the mention m w.r.t. the generic word w is defined as:

$$\mathcal{S}(m,w) = \alpha \cdot cos\_sim(m,w) + (1-\alpha) \cdot freq(m) \tag{8.2}$$

where $cos\_sim(m,w)$ is the cosine similarity between the mention m and the word w in the word embedding model $\mathcal{E}$ selected in Sec. 8.2.1. We concentrate on the most important terms, (i) computing the score value only for the top-k most similar mentions and (ii) filtering out the words which are rarely used in the corpus. To do this, we compute the cumulative frequency of $freq(m)$, and we keep only the mentions determining the 95% of the cumulative.[2]

Step 2.2 Suggest the Best Entry Concepts for the New Mention Once $\mathcal{M}$ is synthesised, the most suitable concepts are identified based on three measures, namely GAS (*Generality*, *Adequacy*, and *Specificity*), that estimate the fitness of a concept c for a given mention m.

Generality and Specificity. The *Generality* (**G**) of a mention m measures to which extent the mention's embedding is similar to the embeddings of all the words in the taxonomy $\mathcal{T}$ as a whole, despite the concept. Conversely, the *Specificity* (**S**) between the mention m and the concept c measures to which extent the mention's embedding is similar to the embeddings that represent the words associated with concept c in $\mathcal{E}$. They are defined as follows.

$$\mathbf{G}_m = \frac{1}{|\mathcal{W}|} \sum_{w \in \mathcal{W}} \mathcal{S}(m,w) \tag{8.3}$$

$$\mathbf{S}_{m,c} = \frac{1}{|\mathcal{F}(c,w)|} \sum_{w \in \mathcal{F}(c,w)} \mathcal{S}(m,w) \tag{8.4}$$

Adequacy. The *Adequacy* (**A**) between m and c estimates the fitting of the new mention m, extracted from the corpus, to the concept c in the taxonomy, based on the vector representation of m and the words $w \in \mathcal{F}(c,w)$, i.e. their use in the corpus. **A** is computed as:

$$\mathbf{A}_{m,c} = \frac{e^{\mathbf{S}_{m,c}} - e^{\mathbf{G}_m}}{e-1} \in [-1,1] \tag{8.5}$$

[2] k is set to 1,000 whilst α is set to 0.85 to weight the frequency less than the similarity.

On one side, the *Adequacy* of a mention m to the concept c is directly proportional to the similarity with the other words $w \in \mathcal{F}(c, w)$ (i.e., the *Specificity* to the concept c). On the other side, the *Adequacy* is also inversely proportional to the similarity of m with all the words $w \in W$ (i.e., its *Generality*). The *Adequacy* is defined to hold the following:

$$\mathbf{A}_{m,c} \begin{cases} \gtreqless 0 & \text{if } \mathbf{S}_{m,c} \gtreqless \mathbf{G}_m \\ > \mathbf{A}_{m_2,c_2} & \text{if } \mathbf{S}_{m,c} - \mathbf{G}_m = \mathbf{S}_{m_2,c_2} - \mathbf{G}_{m_2} \wedge \mathbf{S}_{m,c} > \mathbf{S}_{m_2,c_2} \end{cases}$$

The first property guarantees zero to act as a threshold value, that is, a negative value of **A** indicates that the mention is related more to the taxonomy, rather than that specific concept c. Conversely, a positive **A** indicates the mention m might be a sub-concept of c. The second property guarantees that, given two pairs of concepts and mentions - e.g. (m, c) and (m_2, c_2) - if the difference between their *Specificity* and *Generality* values is the same, then the pair having the higher *Specificity* will also have a higher value of *Adequacy*, still allowing `NEE` to distinguish between the two.

8.2.3 Step 3: Vote and Enrich

Finally, the method requires some field experts to validate the outcome of Sec. 8.2.1 and Sec. 8.2.2, which are fully automated. The user evaluation is composed of two questions. We ask to evaluate *Q1*) if the mentions extracted from the corpus in Step 2.1 are valid additions to the taxonomy and *Q2*) to which extent the concepts suggested as an entry for a new mention are appropriate for it, based on the name of the mention and the concepts. For *Q1*, the user is asked to give a yes/no answer, while *Q2* is evaluated using a 1-6 Likert scale (from 1: *Completely disagree*, to 6: *Completely agree*). The user feedback is used as judgement for quality, meaning that a high evaluation of the best-proposed suggestion implies a high-quality suggestion.

8.3 Requirements and Hyper-parameters

In order to use `NEE`, the user needs to start by selecting the taxonomy they need to update, and a corpus of data which is relevant for the field, as shown in the first line of PseudoCode 1.

The user also needs to define a grid of hyper-parameters for the training of word embedding models (step 1), and the value of $\alpha \in [0, 1]$ (see Eq. 8.2 in step 2). The greater α is, the more important the cosine similarity between the mention and the word in the taxonomy is, the smaller, the more important the frequency of the mention in the corpus is.

9

Taxonomy Alignment via Word Embeddings

In Chapter 7, we introduced the topic of taxonomy induction and alignment, and we saw how this topic has been dealt with in the literature. In this Chapter, we propose `WETA`, a domain-independent, knowledge-poor method for automatic taxonomy alignment via word embeddings. `WETA` associates all the leaf terms of the origin taxonomy to one or many concepts in the destination taxonomy, employing a scoring function, which merges the score of a hierarchical method based on cosine similarity and the score of a classification task[1].

9.1 Setting the Stage

As seen in Chapter 3, lexical taxonomies are a natural way of expressing the semantic relationships between words and concepts through *is-a* relations. They constitute a lingua franca for the domain they represent. Taxonomies are the mainstay of several downstream

[1] Some results of this Chapter were published in [67].

applications, beyond being used by industry practitioners and policymakers to facilitate knowledge sharing and information retrieval. Usually, within a single domain, multiple taxonomies are built by different institutions for different purposes, and sometimes for several languages or jargon. As a consequence, the problem of mapping those different taxonomies is of primary importance.

The manual mapping of two taxonomies by domain experts is a time-consuming and costly task, which often leads to inaccuracies. Here arises the importance of developing automated taxonomy alignment methods. An automatic taxonomy alignment method allows mapping each concept in the origin taxonomy onto a ranked list of the most relevant concepts in the destination taxonomy, helping the domain experts reduce the choices to a smaller set of concepts.

To develop our automated taxonomy alignment method, we resort to word embeddings, which have empirically shown to capture linguistic regularities from texts [107], demonstrating their ability to enrich existing knowledge structures as well, see e.g. [125, 51].

Note that `WETA` is knowledge-poor and domain-independent since it doesn't require any resource but the two taxonomies.

Suppose we have two taxonomies, $\mathcal{T}_o$ and $\mathcal{T}_o$, defined as in Def. 3.1.1 in Chapter 3. Given a hierarchical level x, we define $\mathcal{C}_x^o$ and $\mathcal{C}_x^d$ as the set of all the $c \in \mathcal{T}_o$ and $c \in \mathcal{T}_d$ respectively, that are classified in level x.

Definition 9.1.1 (Taxonomy Alignment Problem (TAP)) *Given an origin taxonomy $\mathcal{T}_o$ and a destination taxonomy $\mathcal{T}_d$, the problem is to suggest one or more concepts $c \in \mathcal{T}_d$ for each word $w \in \mathcal{T}_o$. A Taxonomy Alignment Problem (TAP) consists in assigning to each $w \in \mathcal{T}_o$ n possible matches $c \in \mathcal{T}_d$.*

Definition 9.1.2 (`WETA`) *Let $\mathcal{T}_o$ and $\mathcal{T}_d$ be respectively an origin and a destination taxonomy as in Def.3.1.1. `WETA` provides a 3-tuple $(\psi, h, \mathcal{S})$, where:*

- *$\psi : \mathcal{W}^o \times \mathcal{C}^d \rightarrow [0,1]$ is a scoring function that estimates the relevance of $c \in \mathcal{T}_d$ with respect to a word $w \in \mathcal{T}_o$ considering the prediction scores of a multi-class classification task;*
- *$h : \mathcal{W}^o \times \mathcal{C}^d \rightarrow [0,1]$ is a scoring function that estimates the relevance of $c \in \mathcal{T}_d$ with respect to a word $w \in \mathcal{T}_o$ considering the semantic similarity of w with c and all its hypernyms;*

- $\mathcal{S}(\psi, h) \subseteq \{(w, c) \mid w \in \mathcal{W}^o \wedge c \in \mathcal{C}^d\}$ *is the score of an alignment relation existing between a word in* $\mathcal{T}_o$ *and a concept in* $\mathcal{T}_d$*, blending the results of the above-mentioned scoring functions.*

9.2 Methodology

This section aims to describe the global approach used to align the taxonomies $\mathcal{T}_o$ and $\mathcal{T}_d$., which is mainly composed by the steps shown in Fig.9.1.

The first step allows us to train and select the best word embedding model, which is then used in the second step to suggest for each leaf concept $w_o \in \mathcal{W}^o$ n possible alignments $c_d \in \mathcal{C}_p^d$. The last step consists of the suggestions validation because the utility of `WETA` is the help it provides to the domain experts, narrowing the choices for the alignment that would otherwise be done from scratch.

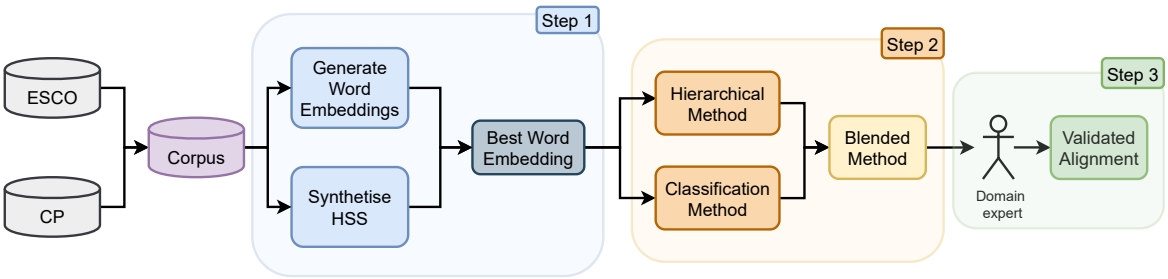


Figure 9.1 A representation of the `WETA` *workflow highlighting its main modules.*

The structure of `WETA` is shown in PseudoCode 2.

9.2.1 Step 1: Generate and Evaluate Embeddings

The main goal of the first step of `WETA` is to induce a vector representation of taxonomic terms, such that it represents as much as possible the similarity of words within the taxonomy. To accomplish this, we apply the procedure described in Chapter 5. We perform three distinct tasks:

- We generate word embeddings;
- We compute the `HSS` (see Section 5.2.1 in Chapter 5) for terms in $\mathcal{T}_o$ and $\mathcal{T}_d$;

- We select the embeddings for which the correlation between the cosine similarity between taxonomic terms and their HSS is maximised for both $\mathcal{T}_o$ and $\mathcal{T}_d$ (see Section 5.2.3 in Chapter 5).

Embeddings Generation

As the input for the word embedding generation, we consider a corpus constituted of one sentence for each concept in $\mathcal{T}_o$ and $\mathcal{T}_d$, containing the label of the concept, all the words associated with it and its description.

For the generation of the word embedding models, WETA can rely on state-of-the-art methods for word embedding generation. Their use allows for the testing of different vector space models, tuning the parameters to find the best vector space to represent the taxonomies.

HSS Computation

We compute the HSS between all the pairs of words in the taxonomy. We defined the HSS in Section 5.2.1 in Chapter 5 as:

$$sim_{\texttt{HSS}}(w_1, w_2) = \sum_{\ell \in \mathcal{L}} \hat{p}\left(\ell = LCA \mid w_1, w_2\right) \times IC(LCA) \tag{9.1}$$

between two words w_1 and w_2.

Selection of the Best Word Embedding

The performance of each word vector model generated before is assessed - following the TaxoVec framework (see 5.2.3) - by the Pearson Correlation of the HSS between all the pairs of words in the taxonomy with the cosine similarity between their vectors in the model space.

9.2.2 Step 2: Taxonomy Alignment Method

We propose a methodology for taxonomy alignment that suggests, for each word, or leaf concept, $w_o \in \mathcal{T}_o$, a set of n possible destination concepts in $\mathcal{T}_d$. The destination concepts are selected among most specialised concepts in $\mathcal{T}_d$, i.e. those which are at the lowest level p, that is $\{c_1, \ldots, c_n\} \in \mathcal{C}_p^d$.

PseudoCode 2 `WETA`

Require: $\mathcal{T}_o(\mathcal{C}^o, \mathcal{W}^o, \mathcal{H}^c, \mathcal{F})$, $\mathcal{T}_d(\mathcal{C}^d, \mathcal{W}^d, \mathcal{H}^c, \mathcal{F})$ as in Def. 3.1.1;

$\mathcal{E} \leftarrow$ best_embedding$(\mathcal{T}_o, \mathcal{T}_d)$ } Generate and evaluate embeddings

$L_{Hp}^{(w_o)} \leftarrow \varnothing$

for all $w_o \in \mathcal{W}^o$ **do**

  $L_H^{(w_o)} \leftarrow$ compute Eq. 9.2

  $L_H^{*(w_o)} \leftarrow$ compute Eq. 9.3

  $L_{Hp}^{(w_o)} \leftarrow$ compute Eq. 9.4

} Hierarchical approach

$L_C^{(w_o)} \leftarrow \varnothing$

for all $w_o \in \mathcal{W}^o$ **do**

  $\{c_{1(p)}, \ldots, c_{n(p)}\} \in \mathcal{C}_p^d \leftarrow$ classification_task$(\mathcal{T}_o, \mathcal{T}_d)$

  $L_C^{(w_o)} \leftarrow$ compute Eq. 9.5

} Classification approach

$\mathcal{S} \leftarrow \varnothing$

for all $w_o \in \mathcal{W}^o$ **do**

  $\mathcal{S} \leftarrow u$ common matches from $L_{Hp}^{(w_o)}, L_C^{(w_o)}$

  $\mathcal{S} \leftarrow \mathcal{S} \cup$ best $\lceil \frac{n-u}{2} \rceil$ suggestions from $L_{Hp}^{(w_o)} \setminus \mathcal{S}$

  $\mathcal{S} \leftarrow \mathcal{S} \cup$ best $\lfloor \frac{n-u}{2} \rfloor$ suggestions from $L_C^{(w_o)} \setminus \mathcal{S}$

} Blended approach

return $(\mathcal{W}^o, \mathcal{C}^d, \mathcal{S})$

To do this, we perform two different processes that lead to independent results, and then we blend their suggestions to obtain a robust mapping between taxonomies.

Hierarchical Approach

For each $w_o \in \mathcal{W}^o$, the set of words of the origin taxonomy, we create a set of tuples that contains the cosine similarity between w_o and each element in $w \in \mathcal{W}^d$:

$$L_H^{(w_o)} = \{(w, sim(w_o, w)) \mid w \in \mathcal{W}^d\} \ \forall w_o \in \mathcal{W}^o \tag{9.2}$$

where *sim* is the cosine similarity between the vector representations of the two inputs in the best word embedding model, see Sec. 9.2.1. This set is then ordered decreasingly on $sim(w_o, w)$.

Given the i-th pair $(w, sim(w_o, w))_i$, we can refer to its similarity as sim_i to define the function W as follows:

$$W(sim_i) = (sim_i)^2 \cdot (sim_i - sim_{i+1}) \tag{9.3}$$

where to transform each similarity score, we consider the next similarity in the ordered set of tuples. Thanks to W, we can highlight the situations in which, for example, the similarity score between w_o and the first word in $L_H^{(w_o)}$ is significantly higher than the other scores, rather than a situation where all the elements present high similarity.

Now, we exploit the hierarchical concepts: for each $w \in L_H^{(w_o)}$, we extract its respective hypernym at the level p. In other words, we consider the more specific concept level for each word in $L_H^{(w_o)}$ because our goal is to map the words in $\mathcal{T}_o$ into concepts in $\mathcal{T}_d$, even though we started the procedure considering the similarities word-word. We define $L_{Hp}^{(w_o)}$ as the set that contains all the level p hypernyms of every $(w, sim) \in L_H^{(w_o)}$, we order them, and we keep the n with associated the highest similarity. More formally:

$$\begin{aligned} L_{Hp}^{(w_o)} &= \{(c_p, h_p) \mid \exists (w, sim) \in L_H^{(w_o)} : \mathcal{F}(c_p, w) \wedge c_p \in \mathcal{C}_p^d\} \ \forall w_o \in \mathcal{W}^o \\ &\text{with } h_p = \max_{sim \in S_p} sim,\ S_p = \{sim \mid (w, sim) \in L_H^{(w_o)}\} \end{aligned} \tag{9.4}$$

We keep only the n pairs with the highest similarity score h_p.

Classification Approach

This approach relies on a multi-class classification task, developed at the level p of $\mathcal{T}_d$.

For the classification task, we have $\mathcal{C}_p^d$ as the target variable, which is the more specific concept level that does not represent an entity, and the word embeddings associated with each $w_o \in \mathcal{W}^o$ as the independent variable. For each word w_o, at the end of the classification process, we consider the prediction scores and select the n top-scored level p concepts:

$$L_C^{(w_o)} = \{(c_{p(1)}, \psi(c_{p(1)})), \ldots, (c_{p(n)}, \psi(c_{p(n)}))\} \ \forall w_o \in \mathcal{W}^o \tag{9.5}$$

where $\psi(c_{p(i)})$ is the prediction score of $c_{p(i)}$ associated to $c_{p(i)}$ for the instance w_o.

Blended Approach

The last part of this step blends the results obtained respectively from the hierarchical approach and the classification one.

First, for each $w_o \in \mathcal{W}^o$, we store the u shared matches of the two methods. We complete the sets with other $r = n - u$ matches by considering some suggestions from the hierarchical approach and some from the classification one. Since the latter allows us to obtain a better final performance, we choose $\lceil \frac{r}{2} \rceil$ suggestions from the classification approach's set and $\lfloor \frac{r}{2} \rfloor$ from the hierarchical approach's one.

9.2.3 Step 3: Evaluation of the Suggestions

The usefulness of `WETA` lies in providing a limited number of suggestions to the domain experts to simplify their work of taxonomy alignment that otherwise would be all manual. The last step consists of the validation by the domain experts of the suggestions provided to complete the alignment procedure.

To evaluate the methodology based on the experts' validation, we consider the top-n Accuracy, which lets us compute how many of the n suggestions include the concept that has been chosen as the correct one by the domain experts.

For the evaluation, we also use the MRR (Mean Reciprocal Rank) because the taxonomy alignment solution can be seen as a ranking problem since the aim is to provide suggestions to the domain experts to facilitate their work in mapping taxonomies:

$$MRR = \frac{1}{|\mathcal{W}^o|} \sum_{i=1}^{|\mathcal{W}^o|} \cdot \begin{cases} \frac{1}{rank_i} & \textit{if there is a correct suggestion for } i \\ 0 & \textit{otherwise} \end{cases} \tag{9.6}$$

where $rank_i$ refers to the rank position of the correct suggestion.

Lastly, we consider the *wMRR* [16], a weighted version of the *MRR* that also considers the hypernyms of the suggestions to assess the correctness of the suggestion:

$$wMRR = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} \cdot \begin{cases} \frac{1}{rank_i} & \textit{if there is a correct suggestion for } i \\ \frac{3}{4}\frac{1}{rank_i} & \textit{if suggestion } i \textit{ has a correct level 3 hypernym} \\ \frac{2}{4}\frac{1}{rank_i} & \textit{if suggestion } i \textit{ has a correct level 2 hypernym} \\ \frac{1}{4}\frac{1}{rank_i} & \textit{if suggestion } i \textit{ has a correct level 1 hypernym} \\ 0 & \textit{otherwise} \end{cases} \tag{9.7}$$

9.3 Requirements and Hyper-parameters

In order to use `WETA`, the user needs to start by selecting the two taxonomies they need to bridge, as shown in the first line of PseudoCode 2.

The user also needs to define a grid of hyper-parameters for the training of word embedding models for step 1. For step 2, they need to choose a value n, the number of suggestions for each word, or leaf concept, $w_o \in \mathcal{T}_o$, and the methodology to use for the classification task. In Chapter 12 for example we considered $n = 5$ and we used 5-fold cross-validation to select the most performant out of four different classification methods: Random Forest, Support Vector Machine, K-Nearest Neighbours, and a 2-layer Neural Network.

Part IV

Word Embeddings and Taxonomies in Labour Market Intelligence

10

Setting the Stage for Labour Market Intelligence

In this concluding part, we will present various applications of the methodologies introduced in the second and third segments of this thesis, focusing on a specific and pertinent domain: Labour Market Intelligence (LMI).

10.1 What is Labour Market Intelligence

Recently, the labour demand and supply conveyed through specific Web portals and services have grown exponentially. That contributed to introducing the term *Labour Market Intelligence* (LMI), which refers to the use and design of AI algorithms and frameworks to interpret Labour Market Information for supporting decision-making (see, e.g., [142, 115, 82, 143, 147, 98, 159, 42]).

In the LMI scenario, the problem of monitoring, analysing, and understanding these labour market changes (i) timely and (ii) at a very fine-grained geographical level has become practically significant in our daily lives. Machine learning has been applied to compute and

estimate the impact of robotisation on the occupations of the US Labour Market [61, 60] as well as to investigate skill relevance in the US standard taxonomy O*NET [3].

In 2010, the European Commission issued the communication “A new impetus for European Cooperation in Vocational Education and Training (VET) to support the Europe 2020 strategy",[1] aimed at promoting education systems in general, and VET in particular. In 2016, the European Commission highlighted the importance of Vocational and Educational activities, as they are "valued for fostering job-specific and transversal skills, facilitating the transition into employment and maintaining and updating the skills of the workforce according to sectoral, regional, and local needs".[2] In 2016, the EU and Eurostat launched the ESSnet Big Data project, involving 22 EU member states with the aim of "integrating big data in the regular production of official statistics, through pilots exploring the potential of selected big data sources and building concrete applications". In 2014, the EU Cedefop agency - aimed at supporting the development of European Vocational Education and Training - launched a call-for-tender[3] for realising a system able to collect and classify Web Job Advertisements from 5 EU Countries (see [38]). Moreover, a further project started to extend and scale the ML-based system to the whole EU, including its 27 country members plus the UK and all the 32 languages of the Union [39] (see some results e.g. [26, 27, 47, 29, 44]). The two tools presented in Chapters 11 and 12 are framed within this research activity.

Labour Market Intelligence constitutes a valuable field in which we can apply the methodologies developed and explained in the previous Chapters of this thesis for multiple reasons. Firstly, in the context of the European Labour Market, there is a dominant taxonomy, ESCO, the European multilingual classification of Skills, Competencies, and Occupations (see Section 10.2). This taxonomy is recognised as the standard in the European context, and it needs to be kept up to date and to be able to interact with the national taxonomies that were used previously. Secondly, in light of what we exposed at the beginning of this Chapter, there is a

[1] Publicly available at https://goo.gl/Goluxo

[2] Commission Communication "A New Skills Agenda for Europe" COM(2016) 381/2, available at https://goo.gl/Shw7bI

[3] Real-time Labour Market information on Skill Requirements: Setting up the EU system for Online Job Advertisements analysis AO/DSL/VKVET-GRUSSO/Real-time LMI 2/009/16. Contract notice - 2016/S 134-240996 of 14/07/2016

large amount of data on this topic, Online Job Ads (OJAs), which can be used to train word embedding models able to represent all this information gathered from the web.

10.1.1 The Web Intelligence Hub Online Job Advertisements Database

The data used in the works described in the following chapters is part of the Web Intelligence Hub Online Job Advertisements database (WIHOJA) [4]. The data consists of the Online Job Advertisements (OJA) collected and classified in the EU, including its 27 country members plus the UK, from 2018 onward. For different works, we considered the data referred to some of these countries and in various timelines - coherently with the scope and the availability.

10.2 ESCO Taxonomy

In the context of LMI, multiple taxonomies define occupations and skills, and their main difference is the context in which they are applied.

In the USA context, O*NET is the standard taxonomy[5], while ESCO (European Skills, Competences, Qualifications, and Occupations)[6] is the European multilingual standard classification of Skills, Competences, and Occupations. It acts as a dictionary, describing, identifying, and classifying professional occupations and skills relevant to the EU labour market and education and training. Those concepts and the relationships between them can be understood by electronic systems, and that allows different online platforms to use ESCO for services like matching job-seekers to jobs based on their skills, and suggesting training to people who want to re-skill or up-skill.

ESCO provides descriptions of 3008 occupations and 13.890 skills linked to these occupations, translated into 28 languages (all official EU languages plus Icelandic, Norwegian, Ukrainian, and Arabic).

In the works presented in this section, we will always refer to ESCO as the standard taxonomy used to classify skills and occupations.

[4] https://www.cedefop.europa.eu/en/about-cedefop/public-procurement/towards-european-web-intelligence-hub-european-system-collection-and-analysis-online-job

[5] https://www.onetonline.org

[6] https://tinyurl.com/sv4squr

11

Tools for Taxonomy Enrichment with New Emerging Occupations and Skills

In this Chapter, we explore two tools, NEO and NES, which aim at automatically enriching the European Skills, Competences, Qualifications, and Occupations taxonomy (ESCO) with new terms from a free text corpus[1]. The taxonomy enrichment procedure is based on the one - which can be applied to any domain - proposed in Chapter 8, and the word embedding selection procedure is based on the one introduced in Chapter 5.

NEO and NES are framed within the research activity of the EU grant collecting and classifying OJAs over all 27+1 EU Countries described in Chapter 10 [53].

[1] Some results of this Chapter were published in [69].

11.1 NEO: Taxonomy Enrichment with New Emerging Occupations

NEO is a framework that (i) suggests new emerging occupations from Online Job Ads (OJAs) along with the most similar concept within the taxonomy, by employing word-embedding algorithms; (ii) proposes GASC measures (Generality, Adequacy, Specificity, Comparability) to estimate the adherence of the new occupations to the most suited taxonomic concept, enabling the user to approve the suggestion and to inspect the skill-gap.

The experiments we performed on 2M+ real OJAs collected in the UK in 2018, sustained by a user study, confirm the usefulness of NEO for supporting the taxonomy enrichment task with emerging jobs.

NEO's workflow is shown in Fig. 11.1.

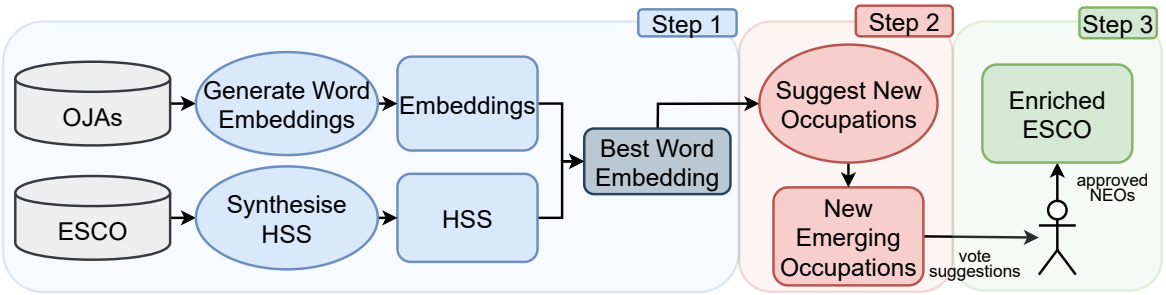


Figure 11.1 A representation of the NEO workflow highlighting the main modules.

11.1.1 Overview of NEO

NEO is a tool developed to assist experts in *i) intercepting new occupation terms widely used in the job market but not part of ESCO and ii) adding them to the adequate class of the ESCO taxonomy*. Its underlying data, a large corpus of OJAs, have been collected in the context of a research call-for-tender described in Chapter 10.

In this context, without loss of generality, we present the results for the ICT-related occupations in the UK market.

We display an example of the ESCO occupation pillar structure in Figure 11.2. ESCO occupation pillar consists of 5 hierarchical levels. Levels 1 to 4 are concept levels that go

from professional macro groups (level 1) to unit groups (level 4), the lowest level concepts. Level 5 is an entity level, containing professional units, i.e. the detailed job denominations called *narrower occupations*. Each ESCO narrower occupation comes with any number *alternative labels*, placed at level 6, that are different ways to refer to the same professions of their level 5 parents. The new mentions found by NEO would be added at this last level, as seen in Figure 11.2.

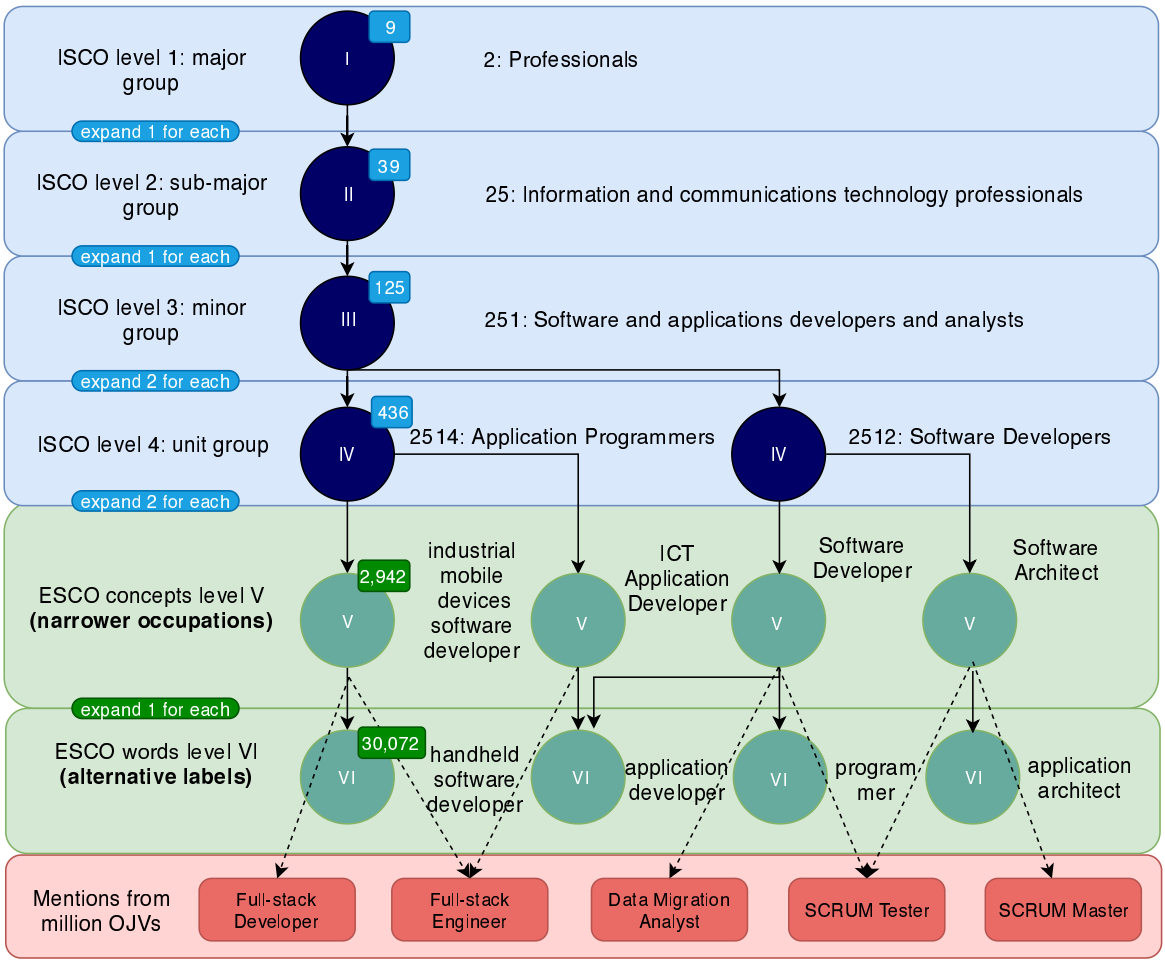


Figure 11.2 Motivating Example. Representation of the ESCO taxonomy, with new mentions representing novel jobs not yet included in ESCO as they emerge from the UK Web Labour Market Demand (2M+ Online Job Advertisements processed in 2018).

Experimental Settings. The corpus contains 2,119,025 OJAs published in the United Kingdom during the year 2018, referring to the ESCO ICT positions reported in Tab. 11.1, and classified as we specified in [26, 28]. OJA's titles were pre-processed, applying the

following pipeline: (1) tokenization, (2) lowercase reduction, (3) punctuation and stop-words removal (4) n-grams computation.

Table 11.1 OJAs collected from the UK in 2018. Only Information and Communication Technology (ICT) occupation codes are shown.

ISCO code	Occupation	Number of OJAs
1330	ICT service managers	176 863
2511	Systems analysts	402 701
2512	Software developers	740 112
2513	Web and multimedia developers	225 784
2514	Applications programmers	30 383
2519	Software and applications developers and analysts	44 339
2521	Database designers and administrators	42 305
2522	Systems administrators	45 542
2523	Computer network professionals	15 411
2529	Database and network professionals	110 210
3511	ICT operations technicians	44 585
3512	ICT user support technicians	168 705
3513	Computer network and systems technicians	55 222
3514	Web technicians	5708
3521	Broadcasting and audiovisual technicians	11 121

11.1.2 Step 1: Synthesise and Select the Word Embedding Model

We trained space vector models using various architectures: *word2vec*, *GloVe* and *fastText*, generating 260 models. Hyper-parameter selection for each architecture was performed with a grid search over the following parameter sets:

- *word2vec* (80 models): algorithm $\in$ {SG, CBOW} $\times$ HS $\in$ {0, 1} $\times$ embedding size $\in$ {5, 20, 50, 100, 300} $\times$ number of epochs $\in$ {10, 25, 100, 200};
- *GloVe* (20 models): embedding size $\in$ {5, 20, 50, 100, 300} $\times$ number of epochs $\in$ {10, 25, 100, 200};
- *fastText* (160 models): algorithm $\in$ {SG, CBOW} $\times$ embedding size $\in$ {5, 20, 50, 100, 300} $\times$ number of epochs $\in$ {10, 25, 100, 200} $\times$ learning rate $\in$ {0.01, 0.05, 0.1, 0.2}

Average training times (with std) in seconds were 890 ± 882 for *word2vec*, 55 ± 74 for *GloVe* and 246 ± 333 for *fastText*, running on an Intel i-7 CPU equipped with 32GB RAM.

The intrinsic evaluation performed to select the embedding that better preserves taxonomic relations follows the framework of `TaxoVec` that we presented in Chapter 5. We computed the Pearson correlation of the cosine similarity between each pair of occupations and their corresponding `HSS`. We did not need to select a sub-sample of the concepts as explained in Section 5.3.2 of Chapter 5 because here we restricted the concepts considered to the ICT ones (see Tab. 11.1) and their narrower occupations, instead of all the ones in the ESCO taxonomy.

The model with highest Pearson correlation, equal to 0.31 with $p\_value \simeq 0$, has the following parameters: architecture=*fastText*, algorithm=CBOW, size=300, epochs=100, learning rate=0.1. Fig. 11.3 provides a scatter plot produced over the best embedding model - as emerges from Tab. 11.1 - generated using UMAP. [2]. Each icon is assigned to one ISCO level 4 group, as in Fig. 11.2. ESCO concepts and words belonging to each group are shown, distinguishing between narrower occupations (shallow shape) and alternative labels (filled shape). Focusing on Fig. 11.3, one might observe that though a *data engineer* and a *data scientist* were designed to be sub-concepts in ESCO, as they belong both to the ▼*2511: System Analyst* ISCO group, their meaning is quite different in the real-labour market, as any computer scientist knows. The former indeed shares much more with ■ *2521: Database designers and administrators* rather than its theoretical group. Conversely, in many practical cases, the taxonomy perfectly adheres to the real labour market demand for occupations. That is the case of ◆ *3521: Broadcasting and audio-visual technicians*, which composes a very tight cluster in the map, showing a perfect match between de-facto and de-jure labour market occupations. That also applies to ∗ *3513: Computer network and systems technicians*, even though to a lesser extent.[3] This analysis is useful to labour market specialists and policymakers to identify mismatches in the taxonomies and provide accurate feedback to improve the taxonomy.

[2] Uniform Manifold Approximation and Projection (UMAP) is a dimension reduction technique that can be used for visualisation similarly to t-SNE, but also for general non-linear dimension reduction

[3] Both best/worst embeddings are available at https://tinyurl.com/worst-neo and https://tinyurl.com/best-neo respectively.

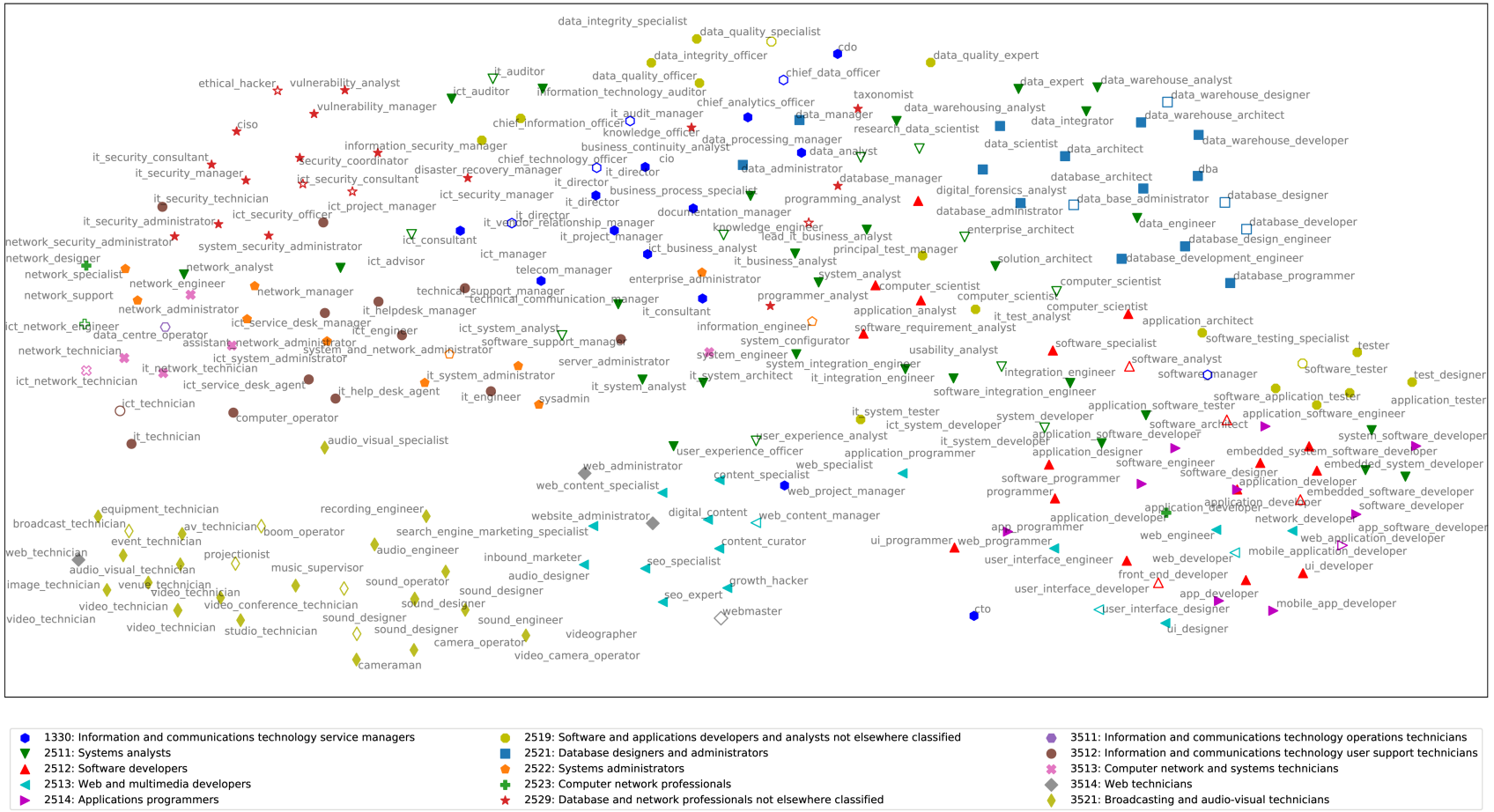


*Figure 11.3 UMAP plot of the **best** word-embedding model resulting in Step 1, that is FastText, CBOW algorithm, Learning rate=0.1, embedding size=100, epochs=100. Each icon is assigned to one ISCO level 4 group, as in Fig. 11.2. The ESCO concepts and words belonging to each group are shown distinguishing between narrower occupations (shallow shape) and alternative labels (filled shape). The image is also available at https://tinyurl.com/best-neo for better visualisation.*

11.1.3 Step 2: Suggest New Emerging Occupations

In Chapter 8, we introduced a set of measures, namely GAS (Generality, Adequacy, and Specificity) to estimate the suitability of a mention $m \in \mathcal{M}$ as an entity of the concepts in $\mathcal{C}$. In the case of NEO, we introduce another measure specific to the Labour Market scenario, the *Comparability*.

Comparability

To better investigate the comparability of the new mention m with the existing concepts in the taxonomy, we consider their required skills. The skills are identified in the context of [40] in the OJAs' descriptions, and classified using the ESCO skills/competencies pillar. Let us consider a set K_c of skills associated with the occupations belonging to the concept c in the OJAs, and a set K_m of skills associated with the mention $m \in \mathcal{M}$ in the OJAs. Given the set $K_U = K_c \cup K_m$ of the L skills associated with at least one out of m and c, we define two L-dimensional vectors $\mathbf{t_c} = (t_{c1}, \ldots, t_{cL})$ and $\mathbf{t_m} = (t_{m1}, \ldots, t_{mL})$ where the generic elements t_{cl} and t_{ml} represent the *revealed comparative advantage* (*rca*) [4] of skill k_l for c and m respectively. If $k_l \notin K_c, t_{cl} = 0$, and similarly if $k_l \notin K_m, t_{ml} = 0$. Given these vectors $\mathbf{t_c}$ and $\mathbf{t_m}$, the *Comparability* ($\mathbf{C}$) between the concept c and the mention m is defined as:

$$\mathbf{C}_{m,c} = \frac{\sum_{l=1}^{L} \min(t_{ml}, t_{cl})}{\sum_{l=1}^{L} \max(t_{ml}, t_{cl})} \tag{11.1}$$

The *Comparability* represents a method to assess the similarity between an ESCO occupation and a potentially new one not based on their vector representation but on their characteristics in the domain of interest. We can consider it together with the first three measures (see Chapter 8): the GASC (Generality, Adequacy, Specificity, and Comparability).

User interface

As a first step, the user selects the starting word w_0 among the occupations already in ESCO (*data analyst* in the example in Fig. 11.4). Then, NEO prompts the five mentions

[4] The $rca \in [0, +\infty]$ was introduced in 2018 in [4] to assess the relevance of skills in the US taxonomy O*Net. We adopted the *rca* to work on ESCO.

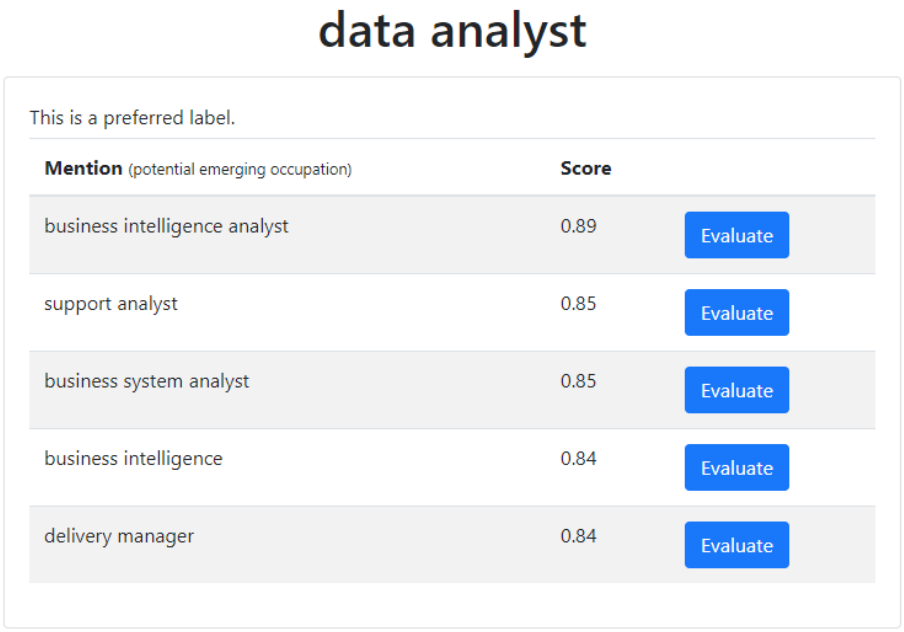
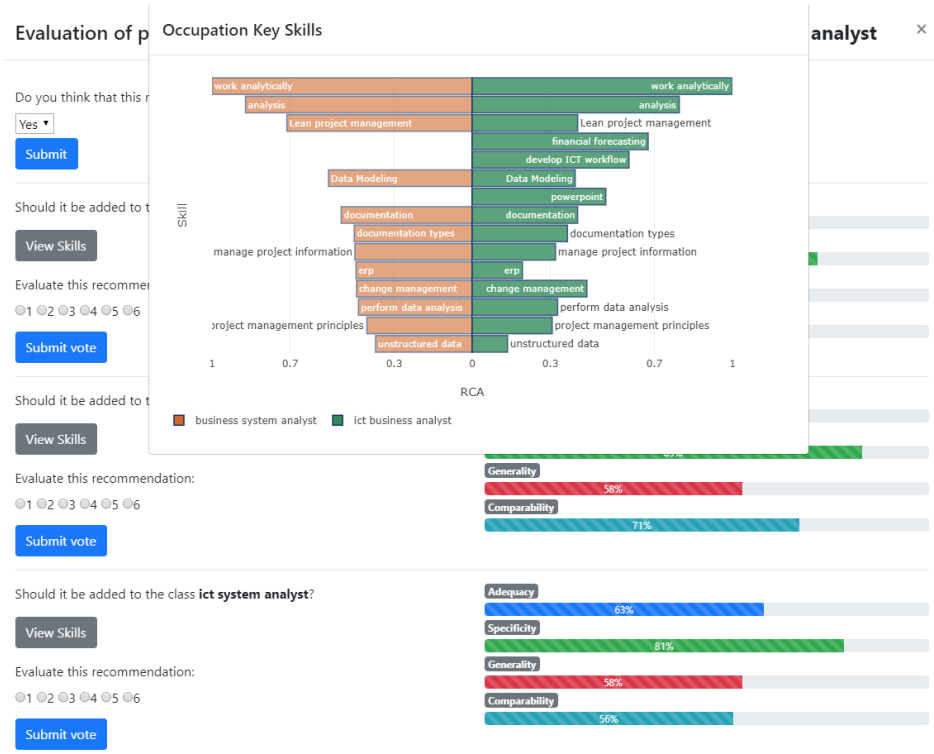


Figure 11.4 New mentions from OJAs starting from the word data analyst.

with associated the highest *score* with w_0 (shown in the example in Fig. 11.4). The user can therefore select a mention m (*business system analyst*) to evaluate to which extent the mention fits as an entity of the starting word's ESCO concept c_j and as an entity of the other two ESCO concepts $c_l, c_k \in \mathcal{C} \setminus c_j$, that is those with associated the highest value of *Adequacy* with the mention m (*ict business analyst* and *ict system analyst* in Fig. 11.5a). For each of these three pairs mention m and ESCO concept, `NEO` provides the GASC measures[5] (see Fig. 11.5a). Specifically, the tool suggests three entry concepts. The first is the starting concept or, in case the user selected a word, the concept to which that word belongs, $\{c_i \in \mathcal{C} | s = c_i \vee \mathcal{F}(s, c_i)\}$. The second and third concepts $c_j, c_k \in \mathcal{C} \setminus c_i$ suggested are the two with the highest value of *Adequacy* A_{ij} with the new mention m, as it is shown in Fig. 11.5a. For each pair, `NEO` provides a comparison of the *rca* of skills for both the mention and the concept (Fig. 11.5b). These skills, together with the GASC measures, support the domain expert in evaluating if the suggested entry is appropriate or not as an entity of a concept, as thoroughly explained in the following Sec. 11.1.4.

[5] In the online tool, all the GASC measures are expressed as a percentage to facilitate the user in the comparison among them.

(a) The suggested classes with their GASC measures.



(b) The new mention's skills and the suggestion's skills.

Figure 11.5 Vote the best class for the new mention selected (business system analyst).

11.1.4 Step 3: Vote and Enrich with User Evaluation

To evaluate the effectiveness of NEO, we recruited 10 ML engineers and labour market experts involved in developing the ML-based system within [40], but not in this research activity. We asked the experts to evaluate the system as detailed in Sec. 8.2.3 in Chapter. 8.

***Q1*: Does NEO suggest valid new emerging occupations?** In *Q1*, we ask the voters whether a suggested mention can be considered an occupation or not. Out of 60 proposed mentions, 11 are repeated, starting from different words. For the remaining 49 unique mentions, 6 of them were evaluated to not be proper occupations, according to the majority of the votes. That means that 88% of the occupations were successfully evaluated as new occupations. Though 6 out of 49 mentions did not pass the test, they are strongly related to the starting concept, referring to skills requested by those job profiles.[6] Fig. 11.6 shows the new occupations found by NEO and the median of Likert scores of experts along with the ESCO concept suggested by NEO and approved by experts.

***Q2*: To which extent do the new mentions fit the suggested taxonomic concepts?** To assess the significance of our GASC measures, we use two well-known hypothesis tests, Spearman's ρ and Kendall's τ coefficient, that proved to be effective in the labour market domain (see, e.g. [156]).

Table 11.2 The results of correlation analysis between GASC and Likert values.

Measure	Kendall τ	p-value $(H_0 : \tau = 0)$	Spearman ρ	p-value $(H_0 : \rho = 0)$
Generality	-0.03	0.14	-0.04	0.13
Adequacy	0.20	2.48×10^{-21}	0.27	1.61×10^{-21}
Specificity	0.14	1.59×10^{-11}	0.19	2.59×10^{-11}
Comparability	0.34	2.21×10^{-60}	0.45	8.33×10^{-62}

The correlation values are shown in Tab. 11.2 while the distribution of the GASC measures grouped according to values of the Likert scale is shown in Fig. 11.7. The association between the Likert values and the corresponding *Adequacy*, *Specificity*, and *Comparability* is positive, and hypothesis tests indicate that it is statistically significant. The strongest correlation is

[6] The mentions evaluated not to be proper occupations are data analytics, business intelligence, penetration testing, operation, data management, and drupal.

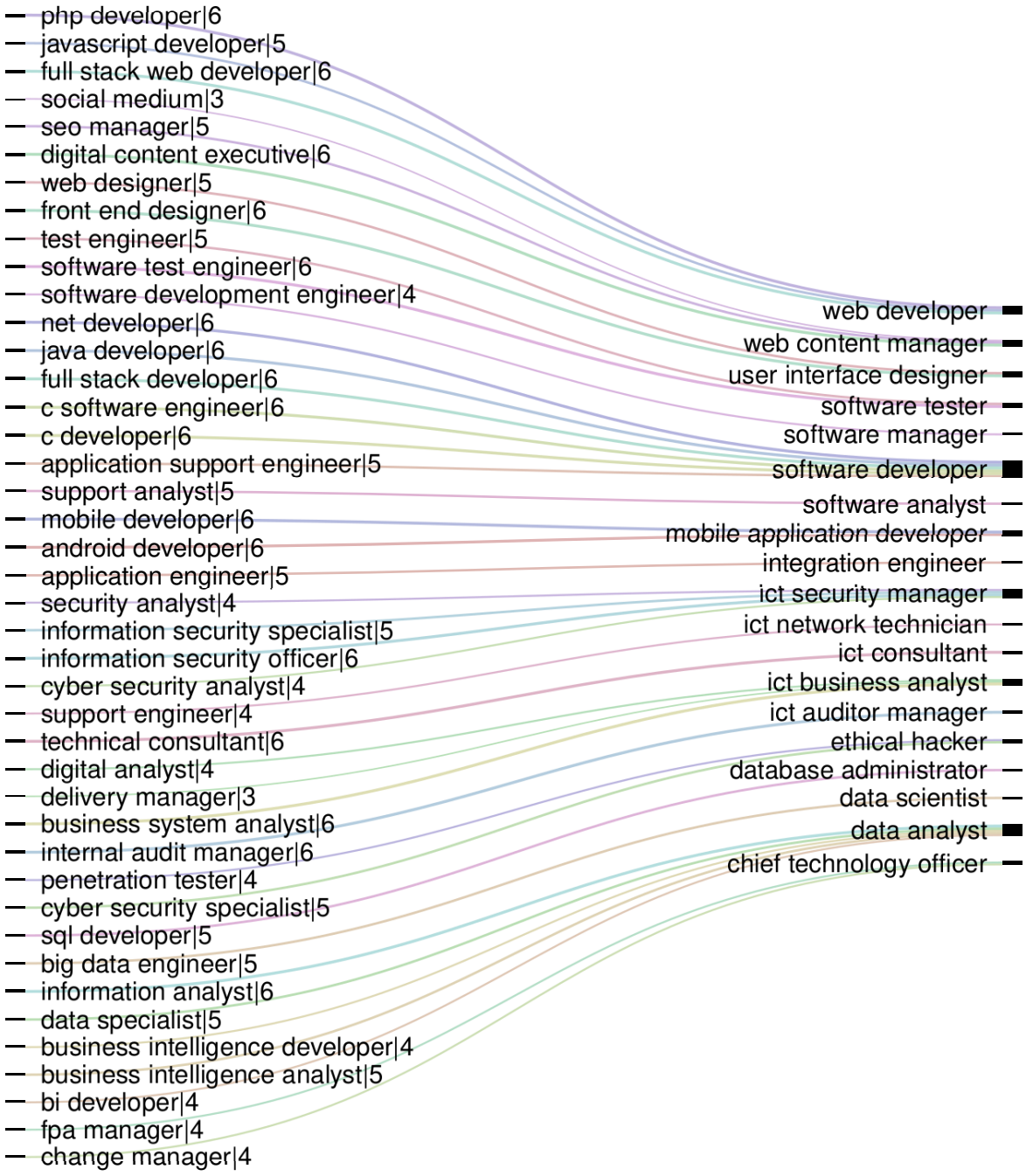
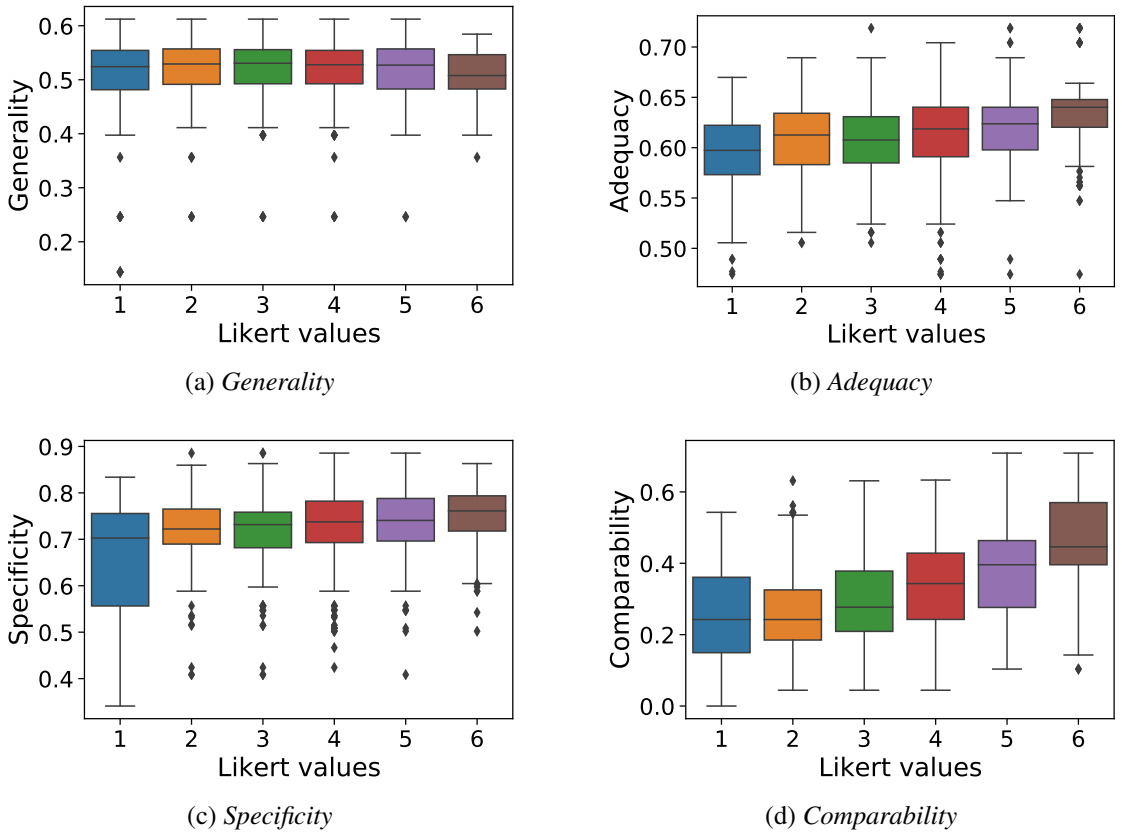


Figure 11.6 Alluvial diagram showing the mentions recognised as New Emerging Occupations with the median of Likert values (i.e., neo|score) and the corresponding ESCO concept suggested by NEO and validated by experts.

between Likert values and *Comparability*, indicating that this is the measure on which the experts relied more. Conversely, the association between the Likert values and the *Generality* is not statistically significant, coherently with the nature of *Generality* that does not aim to rank concepts concerning a mention.



(a) *Generality*

(b) *Adequacy*

(c) *Specificity*

(d) *Comparability*

Figure 11.7 Box-plots representing the distribution of Generality, Adequacy, Specificity *and* Comparability *grouped for each value of the Likert scale.*

In summary, our results - sustained by an expert user study - show that NEO is able (i) to accurately identify novel occupations and (ii) to put them in the right place within the taxonomy. That, in turn, makes NEO a tool for supporting the process of identification of new emerging occupations, enriching the taxonomy accordingly, taking into account the real labour market demand.

11.2 NES: Identifying New Emerging Skills

11.2.1 Overview of NES

NES aims at enriching the European standard labour market taxonomy ESCO [52] with new emerging skills (i.e., skills not included in the ESCO skill pillar yet [90]) derived from Online Job Advertisements (OJAs). It is developed as part of the research activity of an ongoing EU grant aimed at realising the first EU real-time labour market monitoring system [53] (see Chapter 10).

NES is a system that:

- Processes 5M+ OJAs' descriptions published in Italy, France, Spain, Romania and the United Kingdom in their corresponding languages in 2019 and 2020;
- Identifies the terms that might represent a new emerging skill and presents them to the user, along with the most similar ESCO skills and the ESCO occupation groups for which they are required;
- Enables users to approve or disapprove the suggestion.

Our approach is composed of three steps shown in Fig. 11.8 and described in the following subsections: (i) synthesise and select word embeddings, (ii) suggest the new emerging skills, and (iii) confirm the suggestion.

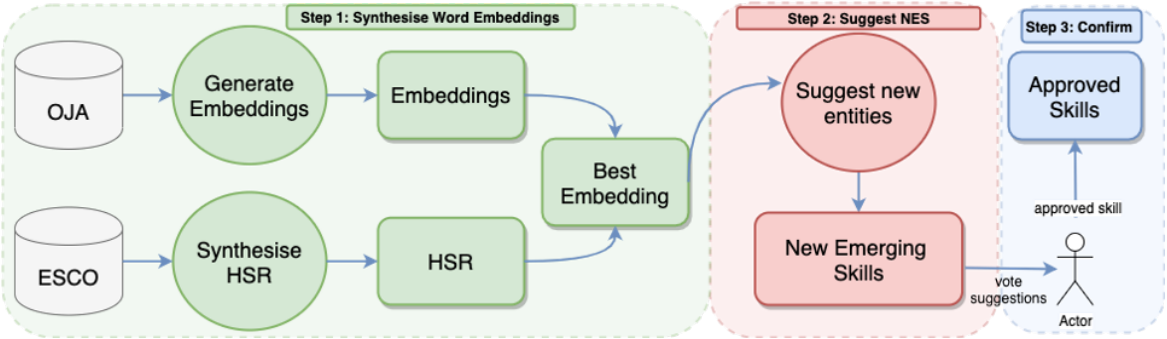


Figure 11.8 Workflow of NES, built following the approach proposed by Giabelli et al. [69].

11.2.2 Step 1: Synthesise Word Embeddings

To prepare the dataset of the OJAs descriptions to be processed by the word embedding algorithm, we need to perform the following steps, which are data-independent but language-dependent:

- Pre-processing. We apply state-of-the-art pre-processing functions that include (i) converting all letters to lower or upper case; (ii) converting numbers into words or removing numbers; (iii) eliminating punctuation; (iv) expanding known abbreviations (e.g., aka, asap); (v) removing stop words, sparse terms, and particular words.
- Selection of the sentinel words for each language. A sentinel word is a word that is more likely to identify a sentence that might contain a skill. The idea is that skills are more likely to appear together in a text. We use the top-100 ESCO skill requested within the OJA corpus as a sentinel word.
- Identification of the skill sentences for each language. Once a list of sentinel words is found, we use them to extract sentences from each OJA description. The idea is to train word embeddings only on sentences containing skills to remove the bias.
- n-gram generation. This step scans the sentences to identify separate words that should be considered as a single word, and we identify up to 4-grams in the pipeline.

At the end of this step, the OJAs descriptions can be seen as a set of *OJA sentences* to be processed by the word-embedding algorithm.

We selected the best one using the HSS and the procedure described in Chapter 5. We trained 24 fastText word vector models with the following architectures and parameters for the French dataset: algorithm $\in$ {SG, CBOW}, size $\in$ {50, 100, 150}, epochs $\in$ {5, 10}, learning rate $\in$ {0.05, 0.1}. For choosing the minimum number of pairs of skills in the taxonomy required for computing the Pearson correlation between the $sim_{\mathtt{HSS}}(w_1, w_2)$ and the cosine similarity for the trained models, we followed the method described in Section 5.3.2. We computed the HSS and the cosine similarity between 30 000 pairs of skills and the best model found, with $r = 0.11$ and $p\_value \simeq 0$, has the following parameters: architecture=*fastText*, algorithm=CBOW, size=50, epochs=10, learning rate=0.1. We used the same parameters also for the models we trained for the datasets in the other languages considered.

To have more homogeneous vector representations, we use the best settings to train a model for each industrial sector, i.e. with only the OJAs referring to that specific sector. The industrial sectors are: *Information and communications*, *Accounting, administration and secretariat*, *Managers*, *Quality, safety, and environmental control and certification*, *Staff management, human resource organisation*, *Installation and maintenance*, *Logistics, transports and distribution*, *Marketing, Communication and Assistance of customers*, *Production of goods, delivery of service*, *Design, research and development*, and *Sales*.

11.2.3 Step 2: Suggest New Skills

This step is aimed at extracting new skills from the corpus of OJAs. Starting from a skill s_0 in the taxonomy $\mathcal{T}$, we consider the top-5 mentions in the corpus $\mathcal{D}$ with associated the highest *score* value $\mathcal{S}$ with s_0, following the Equation 8.2 proposed in Chapter 8. This score considers the cosine similarity and also the frequency of the skills, to filter out the ones less present.

To consider only really novel skills, we discard the mentions that are too similar to the ESCO ones, e.g., n-grams composed by the same words of an ESCO skill in a different order.

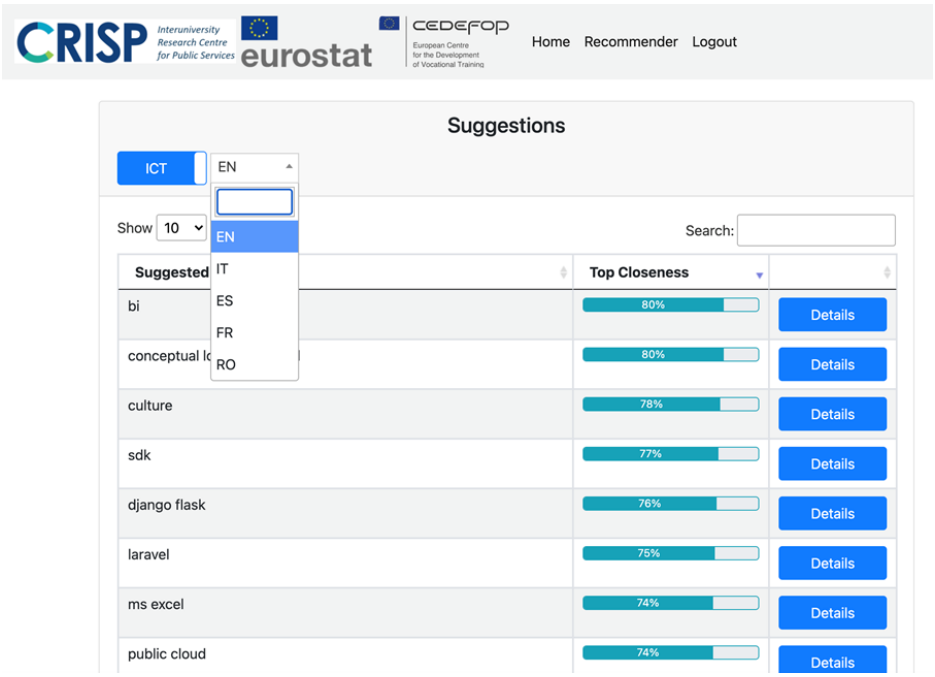
11.2.4 Step 3: Validation

This step validates the outcome of the previous ones - which are fully automated - by asking the following questions to the International Country Experts (ICE).

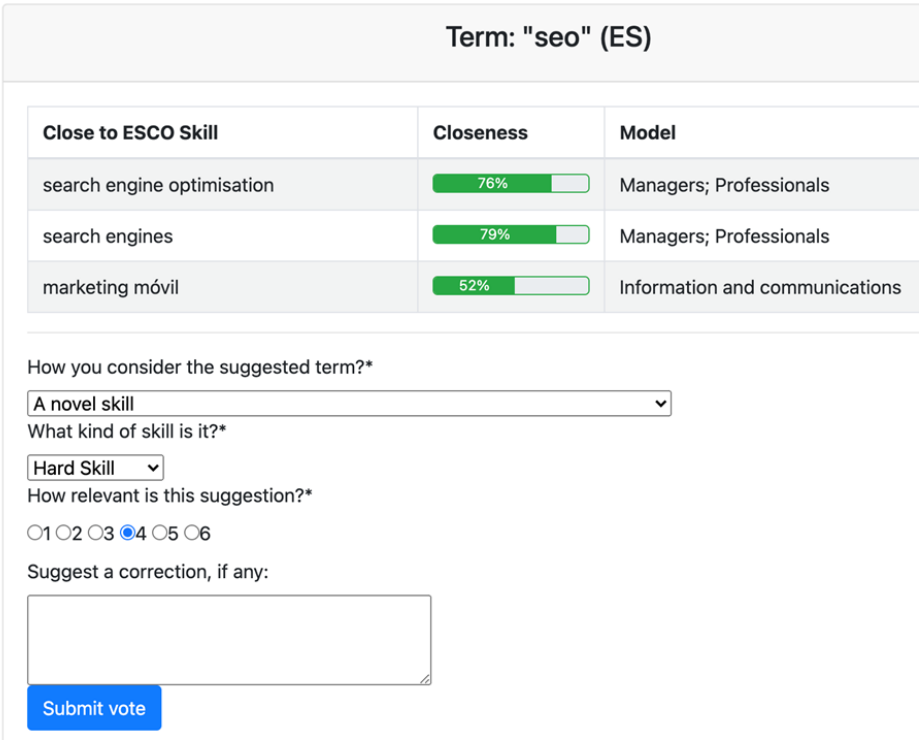
(**Q1**) Do you consider the term:

- A novel skill;
- A specification of an existing one (e.g., MS-Excel is a specification of MS Office);
- A generalisation of an existing one (e.g., MS Office is a generalisation of MS Excel);
- A synonym with the existing one (e.g., ASD is the acronym for Autism Spectrum Disorder);
- The term is neither a novel skill nor an alternative term.

(**Q2**) Do you consider the term a soft/hard/digital skill or none of them?



(a) The list of new skills is suggested as they emerge from the OJA of the selected language.



(b) After selecting a new emerging skill, NES shows which ESCO skills are closest and a form for user feedback.

Figure 11.9 NES user interface.

(**Q3**) Provide a judgement on the relevance of the suggestion on a 6-point Likert scale: the lower, the better. (*1. Strongly Relevant; 2. Relevant; 3. Marginally relevant; 4. Undecided; 5. Not so relevant 6. Not relevant at all.*).

(**Q4**) If you think that the term should be rephrased, please suggest a correction.

The validation framework presented to the ICE is shown in Figure 11.9. In the application, the user can look at the details of each suggestion. For each suggested term, the *top closeness* is the score (see Section 11.2.3) with the existing ESCO skill from which it was found.

An evaluation involving International Country Experts (ICE) from Spain, France, Italy, and Romania[7] to vote the recommendations provided by `NES` shows the system's effectiveness. The results are shown in Tab. 11.3: the accuracy of terms being recognised as skills, whether the expert acknowledged the system's ability in catching specifications, synonym, and generalisations of existing ESCO terms.

Table 11.3 Results from the ICE evaluation.

Country	Accuracy	Specification	Synonym	Generalisation
Spain	85/131 (65%)	✓	✓	✓
France	130/229 (56%)	✓	×	✓
Italy	141/191 (76%)	✓	✓	✓
Romania	41/59 (70%)	✓	✓	✓

[7] UK suggestions are implemented in `NES` but the UK was not involved in the project, thus has no ICE evaluation.

12

JoTA: Aligning Multilingual Job Taxonomies through Word Embeddings

In this Chapter, we present JoTA (Job Taxonomy Alignment), a specific version of WETA (see Chapter 9) used to align the Italian taxonomy of occupations CP[1] and the European ESCO[2] occupation pillar taxonomy. JoTA associates all the leaf terms of the origin taxonomy CP to one or many concepts in the destination taxonomy ESCO, employing a scoring function, which merges the score of a hierarchical method and the score of a classification task[3]. The taxonomy alignment procedure is based on the one - which can be applied to any domain - proposed in Chapter 9, and the word embedding selection procedure is based on the one introduced in Chapter 5.

JoTA is developed in the context of an EU Grant aiming at bridging the national taxonomies of EU countries towards the European Skills, Competences, Qualifications and

[1] http://professioni.istat.it/cp2011/

[2] https://tinyurl.com/sv4squr

[3] Some results of this Chapter were published in [66].

Occupations taxonomy (ESCO) using AI Algorithms[4]. Results, validated within the EU project activities to bridge the Italian Occupation taxonomy CP, confirm the usefulness of WETA in supporting the automatic alignment of national labour taxonomies. The final mapping is available on the ESCO website [5] and it is employed on the EURES Portal[6] to bridge Italian Job Ads to the ESCO taxonomy to obey the EURES 2016/589 regulation.

12.1 Overview of the Data

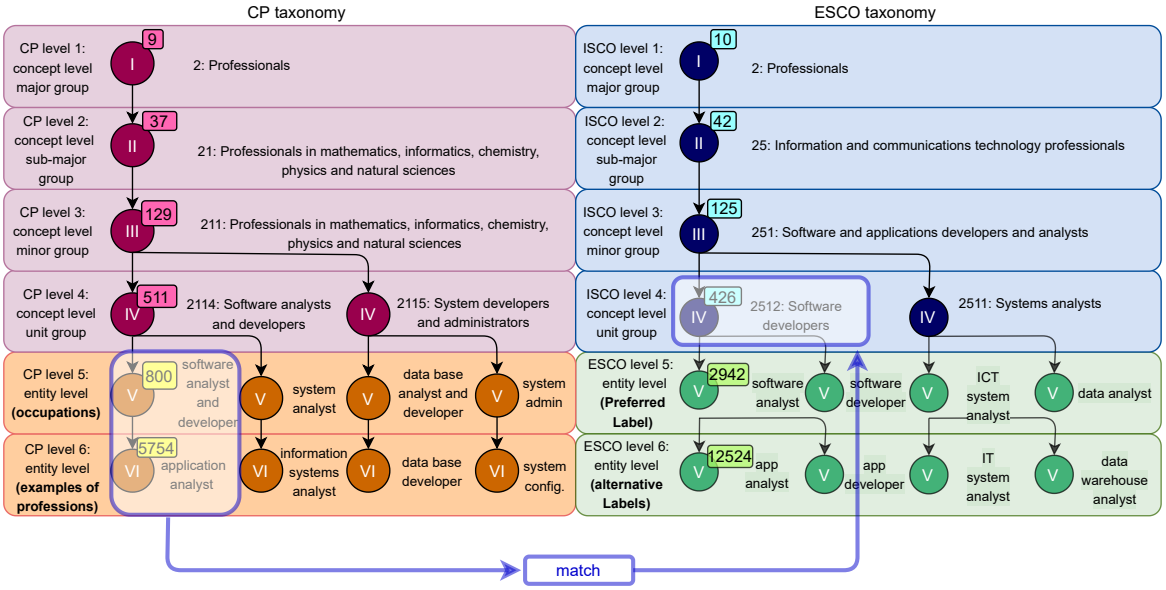


Figure 12.1 Representation of CP and ESCO occupation pillar with examples from ICT professions and a match found by WETA.

Nominally, both the CP and the ESCO occupation pillar consist of 5 hierarchical levels. Levels 1 to 4 are concept levels, as shown in Fig 12.1. They go from professional macro groups (level 1) to unit groups (level 4), the lowest level concepts, that represent the unit professions. Level 5 is an entity level, containing professional units, i.e. the detailed job denominations. In ESCO, those professions are named *preferred labels* while they are called *occupations* in CP. Each ESCO preferred label (and each occupation in CP) comes with

[4] A Data Driven Bridge Towards ESCO using AI Algorithms, granted by EURES (call EaSI-EURES VP/2019/010)

[5] https://esco.ec.europa.eu/en/use-esco/eures-countries-mapping-tables

[6] https://ec.europa.eu/eures/public/homepage

a description (or occupational profile) and any number of non-preferred labels, which are called *alternative labels* in ESCO and *examples of occupations* in CP. Those terms are placed at level 6 in both the taxonomies, but actually, they are different ways to refer to the same professions of their level 5 parents. Therefore, both the taxonomies are constituted by four concept levels and two entity levels, in which the second entity level (level 6) is just different ways to express the same professions of the first level. Since the scope of the alignment is to map the lexicon of the Italian taxonomy CP to the occupation pillar taxonomy ESCO, we develop a mapping that aims at matching each of the 800 level 5 occupations in CP into its most suitable hypernym (parent concept) between the 426 ESCO level 4 concepts. By definition, each level 6 CP entity will be an example of occupation of its level 5 CP hypernym, also in the new mapping. The sizes of the two taxonomies are listed in Tab. 12.1.

Fig. 12.1 provides a detailed overview of the two taxonomies and contains some classification and match examples for ICT professions.

Table 12.1 Size of the two taxonomies' levels.

Taxonomy	**Level**					
	1	**2**	**3**	**4**	**5**	**6**
CP	9	37	129	511	800	5754
ESCO	10	42	125	426	2942	12524

Validation Set The training of the classification task and the best classification model selection is performed on a validation set containing 1677 instances $< w_o, c_d >$.

To create this validation set, firstly, we selected the syntactic matches between words in CP and concepts in the ESCO occupation pillar, comparing their labels and defining the correspondence of characters. Secondly, we explored probabilistic matches by exploiting a fuzzy matching technique that uses the Levenshtein Distance to compute the syntactic similarity between strings. Finally, we manually assessed pairings to remove incorrect results.

12.2 Step 1: Generate and Evaluate Embeddings

For the first step, we computed several word embedding models and evaluated them through the method described in Sec. 5.2.3 in Chapter 5.

`JoTA` employs the state-of-the-art method based on neural networks training FastText. As the input for the word embedding generation, we created a corpus constituted of a sentence for each leaf concept in the ESCO occupation pillar and CP, containing the label of the leaf concept, all the words associated with it, and its description. In total, we had 3742 sentences (800 for CP and 2492 for ESCO) with a mean length of 437 ± 192 all in Italian, since CP has only one Italian version and, for ESCO, we used the Italian occupation's names and descriptions.

We have trained 200 FastText models with these parameters:

- *Algorithm* = {cbow, skipgram};
- *Size* = {50, 100, 150, 200, 300};
- *Epochs* = {10, 50, 100, 150, 200};
- *Learning rate* = {0.01, 0.05, 0.1, 0.2}.

All of them had *minCount*= 1, *minN*= 2. The average training time (with std) in seconds is 86±56.

For choosing the minimum number of pairs required for computing the Pearson correlation between the $sim_{\texttt{HSS}}(w_1, w_2)$ and the cosine similarity for the trained FastText models, we followed the method described in Section 5.3.2.

We have recursively generated samples of pairs, increasing the sample size from 100 to 40 000 by 100 in each step. Fig. 12.2 shows in blue the Pearson correlation of the `HSS` score and the cosine similarity of each paired sample, while the orange line indicates the moving average of 10. To define the Point of Stability (POS) - i.e. a point from which the correlation remains within the Corridor of Stability (COS) - we considered a Corridor of Stability of ±0.01. The POS resulted in 25 500, and we decided to consider 30 000 pairs of words to evaluate the best word embedding model.

The best model found computing the Pearson correlation of `HSS` score and cosine similarity of 30 000 pairs of words is the one with parameters: *algorithm*=cbow, *size*= 150,

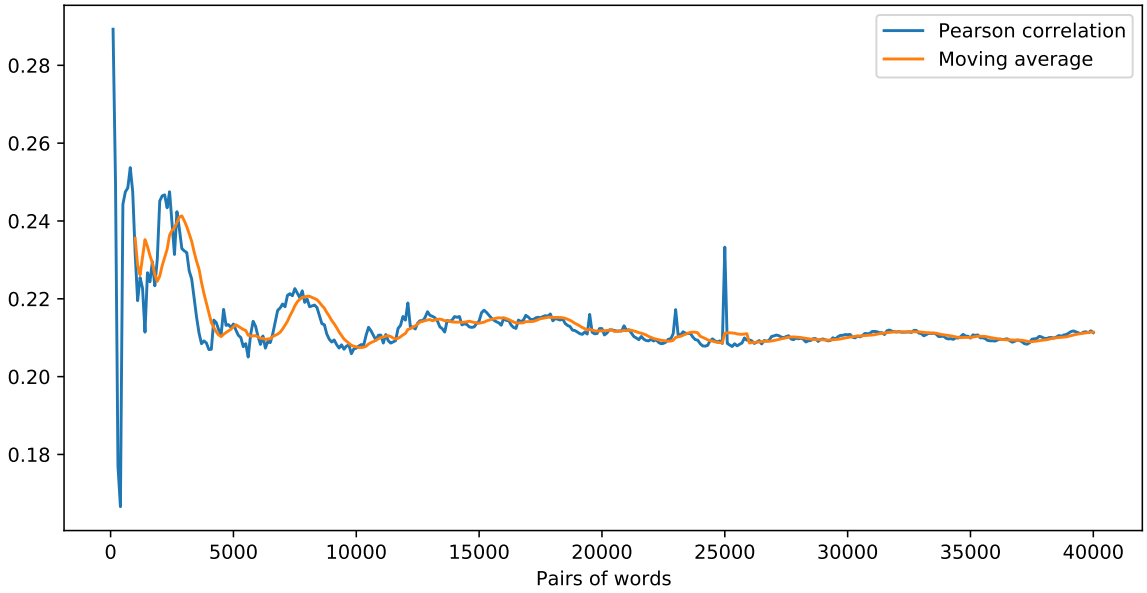


Figure 12.2 Pearson correlation distribution over the samples of pairs of words.

epochs= 100 and *learning rate*= 0.05. With this configuration of parameters, the Pearson correlation is 0.244 with a p-value of $\simeq 0$.

Table 12.2 shows as an example four ESCO occupations and their three most similar vectors in the best word embedding model, selected through the cosine similarity (in parenthesis). We chose these occupations randomly, and from them, it can be seen that the word embedding model seems to capture the semantic similarity in the domain of the labour market.

12.3 Step 2: Classification and Hierarchical Approach

This step is based on the procedure explained in Section 9.2.2 in Chapter 9.

For the classification task, we employed 5-fold cross-validation to tune the hyper-parameters of four different classification methods: Random Forest, Support Vector Machine, K-Nearest Neighbours, and a 2-layer Neural Network. Then we used 5-fold cross-validation to select the network configurations and hyper-parameters with the highest top-5 Accuracy.

The selected model, with a top-5 Accuracy of 0.8867, is a 2-layer neural network with (i) ReLU activation function for the hidden layers, (ii) categorical cross-entropy as loss function, (iii) RMSprop as an optimiser, (iv) 100 epochs, and (v) a batch size of 64.

Table 12.2 Examples of word similarity on the basis of the best word embedding model.

ESCO occupation	Top 3 most-similar	Cosine similarity
armatore_ferroviario (rail layer)	deviatore_ferroviario (rail switchperson)	0.93
	armatore_ferrovie (rail layer)	0.89
	segnalatore_ferroviario (rail marshaller)	0.88
agente_polizia (police officer)	agente_polizia_locale (local police officer)	0.91
	agente_polizia_stato (state police officer)	0.88
	agente_polizia_giudiziaria (court enforcement officer)	0.87
acconciatore_spettacolo (performance hairdresser)	acconciatore_signora (women's hairdresser)	0.80
	acconciatore_uomo (men's hairdresser)	0.80
	acconciatore (hairdresser)	0.73
responsabile_beni_servizi (product and services manager)	responsabile_servizi (services manager)	0.95
	responsabile_servizio_clienti (customer service manager)	0.92
	responsabile_servizi_imprese (business service manager)	0.91

The Hierarchical approach and the Classification approach provide, for each level 5 concept of CP, five concepts in the ESCO occupation pillar that could be used to map the CP concept and that need to be validated by domain experts.

12.4 Step 3: Evaluation

For each $c_o \in \mathcal{T}_o$, we examine their suggested matches with concepts in $\mathcal{T}_d$ to assess the correctness of the method in comparison with the mapping between $\mathcal{T}_o$ and $\mathcal{T}_d$ validated by a group of domain experts involved as reviewers in the project which ended up in December 2020. That, in turn, reduces the human effort for building a mapping from scratch, concentrating the effort on validating the approach.

For the evaluation, we consider the top-5 Accuracy because we are interested in knowing how many of the five suggestions from Step 2 include the concept that has been chosen as the correct one by the domain experts. We also compute the *MRR* (Mean Reciprocal Rank) and the *wMRR* (weighted Mean Reciprocal Rank) that were presented in Section 9.2.3 of Chapter 9.

The evaluation's results are shown in Tab. 12.3.

Table 12.3 The results of top-5 Accuracy, MRR, wMRR.

Method	**top-5 Accuracy**	***MRR***	***wMRR***
String matching	0.4	0.32	0.45
String matching - Token Set	0.49	0.4	0.5
String matching - Token Sort	0.41	0.32	0.44
String matching - Weighted Ratio	0.36	0.28	0.43
Classification Approach	0.77	0.64	0.71
Hierarchical Approach	0.76	0.63	0.69
Blended Approach	0.8	0.66	0.72

We evaluated our final method (*Blended Approach*) and both the single methods that we developed (*Hierarchical* and *Classification Approach*). The classification method achieves better performances on all three evaluation metrics than the hierarchical one, and the blended approach achieves the best performances thanks to the union of the first two methods.

We also evaluated four different methods of string matching, using them as baselines. For these string-matching methods, we relied on a Python package (*RapidFuzz*) that uses the Levenshtein Distance to calculate the differences between strings. All three of our methods achieve better results on the three evaluation metrics than the baseline methods.

In Fig. 12.3, we show the level 1 hypernym of the origin entities and the corresponding level 1 hypernym of the destination concepts. Some categories have the same name (e.g. *Clerical support workers* or *Armed forces*) or an equivalent name (e.g. *Technical professions* and *Technicians and associate professionals*, but there is not an exclusive relationship between them, e.g. a relevant part of the CP *Technical professions* are mapped in the ESCO *Professionals* section. Other concepts have different names, e.g. *Legislators, entrepreneurs, top management* and *Managers*, but the former is aligned almost exclusively with the latter.

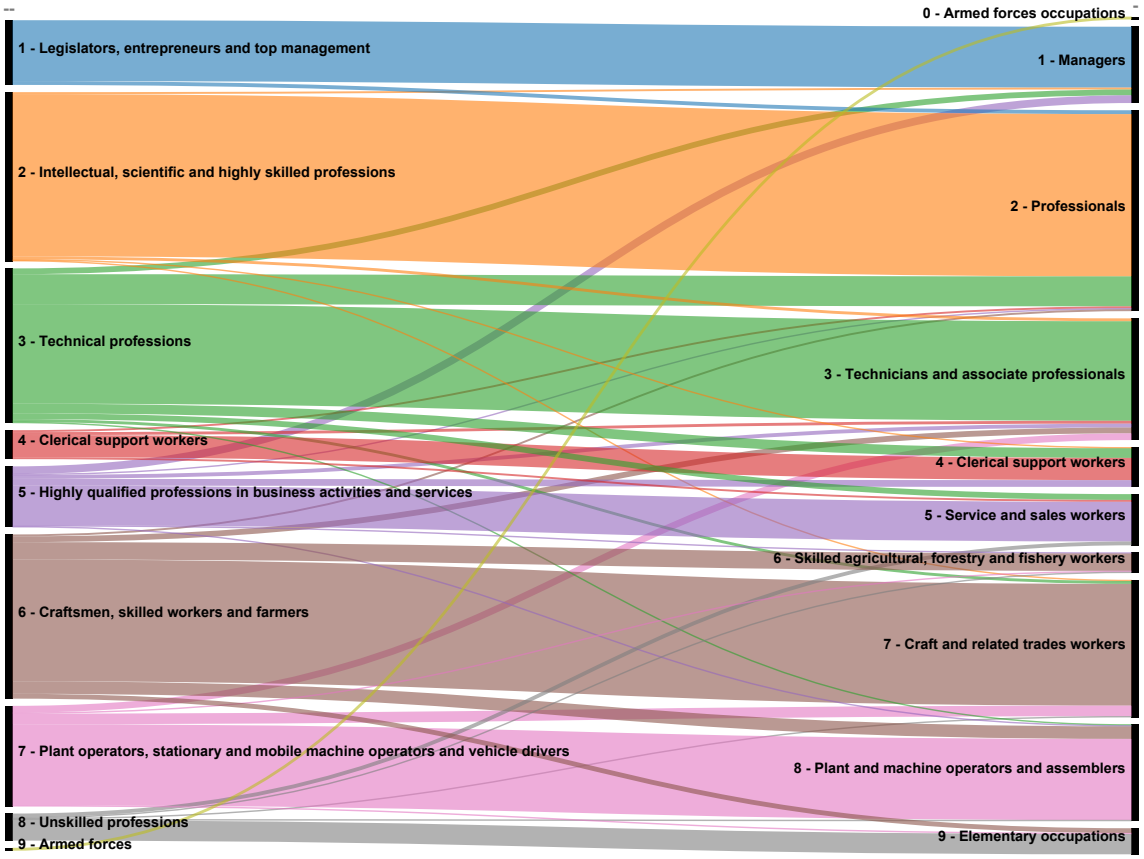


Figure 12.3 Alluvial diagram that depicts the mapping from CP (left-side) to ESCO occupation pillar (right-side) for first levels

13

Conclusion

In this doctoral thesis, we have integrated lexical taxonomies and word embeddings to develop innovative methodologies for improving Natural Language Processing representations. This research has encompassed two primary directions.

Firstly, our work has focused on the intrinsic evaluation of word embeddings, and two novel methods have been designed and developed. The first method, known as `TaxoVec`, involves a framework for selecting taxonomy-aware word embeddings, utilising a measure of taxonomic semantic similarity that we named `HSS`. The second method, `vec2best`, provides a comprehensive evaluation framework for word embeddings, introducing a novel evaluation metric termed *PCE* (Principal Component Evaluation). `vec2best` was developed after `TaxoVec`, reflecting on the fact that it is not always the case that there is a taxonomy relevant to the topic considered. On the other hand, `vec2best` focuses on intrinsic methods of evaluation, and if the embedding models are used only on a specific task, it could be appropriate to evaluate it on that extrinsic task.

Secondly, we have developed two methodologies to enhance and align lexical taxonomies using word embeddings. `NEE` facilitates taxonomy enrichment by estimating the conformity of data to a given taxonomy, thereby identifying previously unrecognised relationships between

Table 13.1 The strengths and weaknesses of the methodologies proposed in this work.

Method	Strengths	Weaknesses
`TaxoVec` [68]	It leverages taxonomic semantic similarity to evaluate word embedding models. It is implemented as a Python package.	It needs a taxonomy to be applied.
`vec2best` [11]	It evaluates word embeddings over different tasks, merging the results through a final metric (*PCE*). It is implemented as a Python package.	It can be applied only to static word embedding models. It does not include extrinsic evaluations.
`NEE` [69]	It leverages word embeddings to identify new entities and concepts to add to a taxonomy from a corpus of texts.	It is reliant on the quality of the corpus of texts.
`WETA` [67]	It is domain-independent and merges a hierarchical method and a classification task for automatic taxonomy alignment.	It is reliant on the quality of the taxonomies (e.g. the quality of the descriptions of the concepts).

entities and concepts. `WETA` represents a domain-independent technique for automatic taxonomy alignment, amalgamating hierarchical similarity and classification tasks into a unified scoring function. These two methodologies have the drawback of being reliant on the data considered, both the taxonomies and the text datasets. For example, `WETA`'s word embeddings are trained on the labels and descriptions of the concepts and words in the taxonomies, and if those were poorly constructed that could affect the performance of the alignment.

In Table 13.1 we summarised the main strengths and weaknesses of these methodologies.

In the final part of this thesis, we have exemplified some practical applications of these methodologies within the context of Labour Market Intelligence. These applications were developed as part of (i) the research activity of an ongoing EU grant aimed at realising the first EU real-time labour market monitoring system [53], and (ii) an EU grant aiming at bridging the national taxonomies of EU countries towards the European Skills, Competences, Qualifications and Occupations taxonomy (ESCO)[1].

[1]A Data Driven Bridge Towards ESCO using AI Algorithms, granted by EURES (call EaSI-EURES VP/2019/010)

13.1 Future Works

Moving forward, our research activities will be focused on two key areas. Firstly, we are committed to expanding `vec2best`, incorporating additional intrinsic evaluation methods such as synonym detection and incorporating other relevant benchmarks for the tasks already included. Furthermore, we are working to include the evaluation method proposed in `TaxoVec` in the `vec2best` framework. We also aim to find a way to include the evaluation of contextual word embeddings, considering tasks that are specific for the evaluation of these types of word embedding models.

Secondly, we are actively exploring the application of the approaches developed in `NEE` and `WETA` to domains distinct from those presented in the concluding section of this thesis.

14

Acronyms

BiLSTM Bidirectional Long Short-Term Memory

CBOW Continuous Bag of Word

COS Corridor of Stability

DL Deep Learning

DSWE Domain-Specific Word Embedding

DAG Directed Acyclic Graph

ESCO European Skills, Competences, Qualifications, and Occupations

HSS Hierarchical Semantic Similarity

ICT Information and Communication Technology

IWE Impact of Word Embedding

k-NN k-Nearest Neighbours

LMI Labour Market Intelligence

LSTM Long Short-Term Memory

ML Machine Learning

MTC Medical Text Classification

NER Named Entity Recognition

NL Natural Language

NLP Natural Language Processing

OJA Online Job Advertisement

PC Principal Component

PCA Principal Component Analysis

PCE Principal Component Evaluation

POS Point of Stability

RS Recommendation System

SA Sentiment Analysis

SG Skip-Gram

SVM Support Vector Machine

TC Text Classification

TM Topic Modelling

UMAP Uniform Manifold Approximation and Projection

WE Word Embedding

List of Figures

List of Tables

Bibliography

[1] Aanen, S. S., Vandic, D., and Frasincar, F. (2015). Automated product taxonomy mapping in an e-commerce environment. *Expert Systems with Applications*, 42(3):1298–1313.

[2] Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., and Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL*, pages 19–27.

[3] Alabdulkareem, A., Frank, M. R., Sun, L., AlShebli, B., Hidalgo, C., and Rahwan, I. (2018a). Unpacking the polarization of workplace skills. *Science Advances*, 4(7).

[4] Alabdulkareem, A., Frank, M. R., Sun, L., AlShebli, B., Hidalgo, C., and Rahwan, I. (2018b). Unpacking the polarization of workplace skills. *Science advances*, 4(7).

[5] Alfarone, D. and Davis, J. (2015). Unsupervised learning of an is-a taxonomy from a limited domain-specific corpus. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1434–1441. AAAI Press.

[6] Allen, C. and Hospedales, T. (2019). Analogies explained: Towards understanding word embeddings. In *International Conference on Machine Learning*, pages 223–231. PMLR.

[7] Almuhareb, A. (2006). *Attributes in lexical acquisition*. PhD thesis, University of Essex.

[8] Aly, R., Acharya, S., Ossa, A., Köhn, A., Biemann, C., and Panchenko, A. (2019). Every child should have parents: a taxonomy refinement algorithm based on hyperbolic term embeddings. *arXiv preprint arXiv:1906.02002*.

[9] Aouicha, M. B., Taieb, M. A. H., and Hamadou, A. B. (2018). Sisr: System for integrating semantic relatedness and similarity measures. *Soft Computing*, 22(6).

[10] Ascari, R., Giabelli, A., Malandri, L., Mercorio, F., and Mezzanzanica, M. (2023). A fistful of vectors: A tool for intrinsic evaluation of word embeddings (manuscript under review). *Cognitive Computation*.

[11] Ascari, R., Giabelli, A., Malandri, L., Mercorio, F., and Mezzanzanica, M. (2024). A fistful of vectors: A tool for intrinsic evaluation of word embeddings (in production). *Cognitive Computation*.

[12] Asudani, D. S., Nagwani, N. K., and Singh, P. (2023). Impact of word embedding models on text analytics in deep learning environment: a review. *Artificial Intelligence Review*, pages 1–81.

[13] Avesani, P., Giunchiglia, F., and Yatskevich, M. (2005). A large scale taxonomy mapping evaluation. In *International Semantic Web Conference*, pages 67–81. Springer.

[14] Bakarov, A. (2018). A survey of word embeddings evaluation methods.

[15] Baker, S., Reichart, R., and Korhonen, A. (2014). An unsupervised model for instance level subcategorization acquisition. In *EMNLP*, pages 278–289.

[16] Bar-Yossef, Z. and Kraus, N. (2011). Context-sensitive query auto-completion. In *Proceedings of the 20th international conference on World wide web*, pages 107–116.

[17] Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*.

[18] Baroni, M., Evert, S., and Lenci, A. (2008). Bridging the gap between semantic theory and computational simulations: Proceedings of the esslli workshop on distributional lexical semantics. *Hamburg, Germany: FOLLI*.

[19] Baroni, M. and Lenci, A. (2011). How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10.

[20] Baroni, M., Murphy, B., Barbu, E., and Poesio, M. (2010). Strudel: A corpus-based semantic model based on properties and types. *Cognitive science*, 34(2):222–254.

[21] Bentivogli, L., Bocco, A., and Pianta, E. (2004). Archiwordnet: integrating wordnet with domain-specific knowledge. In *Proceedings of the 2nd International Global Wordnet Conference*, pages 39–47.

[22] Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

[23] Blair, P., Merhav, Y., and Barry, J. (2016). Automated generation of multilingual clusters for the evaluation of distributed representations. *arXiv preprint arXiv:1611.01547*.

[24] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[25] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

[26] Boselli, R., Cesarini, M., Marrara, S., Mercorio, F., Mezzanzanica, M., Pasi, G., and Viviani, M. (2018a). Wolmis: a labor market intelligence system for classifying web job vacancies. *J. Intell. Inf. Syst.*, 51(3):477–502.

[27] Boselli, R., Cesarini, M., Mercorio, F., and Mezzanzanica, M. (2017a). Using machine learning for labour market intelligence. *ECML PKDD 2017: Machine Learning and Knowledge Discovery in Database*, pages 330–342.

[28] Boselli, R., Cesarini, M., Mercorio, F., and Mezzanzanica, M. (2017b). Using machine learning for labour market intelligence. In *ECML PKDD*, pages 330–342.

[29] Boselli, R., Cesarini, M., Mercorio, F., and Mezzanzanica, M. (2018b). Classifying online job advertisements through machine learning. *Future Generation Computer Systems*, 86:319–328.

[30] Brachman, R. J. (1983). What is-a is and isn't: An analysis of taxonomic links in semantic networks. *Computer*, 16(10):30–36.

[31] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

[32] Bruni, E., Tran, N.-K., and Baroni, M. (2014). Multimodal distributional semantics. *Journal of artificial intelligence research*, 49:1–47.

[33] Camacho-Collados, J. and Navigli, R. (2016). Find the word that does not belong: A framework for an intrinsic evaluation of word vector representations. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 43–50.

[34] Camacho-Collados, J. and Pilehvar, M. T. (2018). From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788.

[35] Camacho-Collados, J., Pilehvar, M. T., Collier, N., and Navigli, R. (2017). Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26.

[36] Cambria, E., Poria, S., Hazarika, D., and Kwok, K. (2018). Senticnet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[37] Caselles-Dupré, H., Lesaint, F., and Royo-Letelier, J. (2018). Word2vec applied to recommendation: Hyperparameters matter. In *RECSYS*.

[38] CEDEFOP (2014). Real-time labour market information on skill requirements: feasibility study and working prototype". https://goo.gl/qNjmrn.

[39] CEDEFOP (2016a). Real-time labour market information on skill requirements: Setting up the eu system for online vacancy analysis. https://goo.gl/5FZS3E.

[40] CEDEFOP (2016b). Real-time labour market information on skill requirements: Setting up the eu system for online vacancy analysis. https://goo.gl/5FZS3E.

[41] Chen, S. W., Wang, S. L., Qi, X. Z., Samuri, S. M., and Yang, C. (2022). Review of ecg detection and classification based on deep learning: Coherent taxonomy, motivation, open challenges and recommendations. *Biomedical Signal Processing and Control*, 74:103493.

[42] Clavié, B. and Soulié, G. (2023). Large language models as batteries-included zero-shot esco skills matchers. *arXiv preprint arXiv:2307.03539*.

[43] Cohen, J. (1992). A power primer. *Psychological bulletin*, 112(1).

[44] Colace, F., Santo, M. D., Lombardi, M., Mercorio, F., Mezzanzanica, M., and Pascale, F. (2019). Towards labour market intelligence through topic modelling. In *Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS)*, pages 5256–5265.

[45] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*. ACM.

[46] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.

[47] Colombo, E., Mercorio, F., and Mezzanzanica, M. (2019). Ai meets labor market: Exploring the link between automation and skills. *Information Economics and Policy*, 47.

[48] Da Silva, J., Revoredo, K., Baião, F., and Euzenat, J. (2020). Alin: improving interactive ontology matching by interactively revising mapping suggestions. *The Knowledge Engineering Review*, 35.

[49] Deng, L. and Liu, Y. (2018). *Deep learning in natural language processing*. Springer.

[50] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[51] Efthymiou, V., Hassanzadeh, O., Rodriguez-Muro, M., and Christophides, V. (2017). Matching web tables with knowledge base entities: from entity lookups to entity embeddings. In *ISWC*, pages 260–277. Springer.

[52] European Commission (2019). ESCO: European skills, competences, qualifications and occupations, available at https://ec.europa.eu/esco/portal/browse.

[53] EUROSTAT (2020). Towards the european web intelligence hub - european system for collection and analysis of online job advertisement data (wih-oja), available at https://tinyurl.com/eurostat2020web. last accessed 15/09/2022.

[54] Euzenat, J., Loup, D., Touzani, M., and Valtchev, P. (2004). Ontology alignment with ola. In *Proc. 3rd ISWC2004 workshop on Evaluation of Ontology-based tools (EON)*, pages 59–68. No commercial editor.

[55] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug).

[56] Faruqui, M. and Dyer, C. (2014). Community evaluation and exchange of word vectors at wordvectors. org. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 19–24.

[57] Fellbaum, C. (1998). *WordNet: An electronic lexical database*. MIT press.

[58] Fellbaum, C., Hahn, U., and Smith, B. (2006). Towards new information resources for public health—from wordnet to medicalwordnet. *Journal of Biomedical Informatics*, 39(3):321–332.

[59] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2001). Placing search in context: The concept revisited. In *WWW*.

[60] Fleming, M., Clarke, W., Das, S., Phongthiengtham, P., and Reddy, P. (2019). The future of work: How new technologies are transforming tasks. *MITIBM Watson AI Lab*.

[61] Frey, C. B. and Osborne, M. A. (2017). The future of employment: How susceptible are jobs to computerisation? *Technological Forecasting and Social Change*, 114(Supplement C):254 – 280.

[62] Fu, R., Guo, J., Qin, B., Che, W., Wang, H., and Liu, T. (2014). Learning semantic hierarchies via word embeddings. In *ACL*, pages 1199–1209.

[63] Ganea, O., Bécigneul, G., and Hofmann, T. (2018). Hyperbolic entailment cones for learning hierarchical embeddings. In *International Conference on Machine Learning*, pages 1646–1655. PMLR.

[64] Geffet, M. and Dagan, I. (2005). The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 107–114.

[65] Gerz, D., Vulić, I., Hill, F., Reichart, R., and Korhonen, A. (2016). Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869*.

[66] Giabelli, A., Malandri, L., Mercorio, F., and Mezzanzanica, M. (2022a). Jota: Aligning multilingual job taxonomies through word embeddings (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36 (11), pages 12955–12956.

[67] Giabelli, A., Malandri, L., Mercorio, F., and Mezzanzanica, M. (2022b). Weta: Automatic taxonomy alignment via word embeddings. *Computers in Industry*, 138:103626.

[68] Giabelli, A., Malandri, L., Mercorio, F., Mezzanzanica, M., and Nobani, N. (2022c). Embeddings evaluation using a novel measure of semantic similarity. *Cognitive Computation*, pages 1–15.

[69] Giabelli, A., Malandri, L., Mercorio, F., Mezzanzanica, M., and Seveso, A. (2020). Neo: A tool for taxonomy enrichment with new emerging occupations. In *International Semantic Web Conference*, pages 568–584. Springer.

[70] Giabelli, A., Malandri, L., Mercorio, F., Mezzanzanica, M., and Seveso, A. (2021a). Neo: A system for identifying new emerging occupation from job ads. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 16035–16037.

[71] Giabelli, A., Malandri, L., Mercorio, F., Mezzanzanica, M., and Seveso, A. (2021b). Skills2graph: Processing million job ads to face the job skill mismatch problem. In Zhou, Z., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4984–4987. ijcai.org.

[72] Gkinko, L. and Elbanna, A. (2023). The appropriation of conversational ai in the workplace: A taxonomy of ai chatbot users. *International Journal of Information Management*, 69:102568.

[73] Gladkova, A., Drozd, A., and Matsuoka, S. (2016). Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *NAACL*.

[74] Hadj Taieb, M. A., Zesch, T., and Ben Aouicha, M. (2020). A survey of semantic relatedness evaluation datasets and procedures. *Artificial Intelligence Review*, 53(6):4407–4448.

[75] Halawi, G., Dror, G., Gabrilovich, E., and Koren, Y. (2012). Large-scale learning of word relatedness with constraints. In *ACM SIGKDD*.

[76] Hanson, S. J. and Bauer, M. (1989). Conceptual clustering, categorization, and polymorphy. *Machine Learning*, 3:343–372.

[77] Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

[78] Hill, F., Reichart, R., and Korhonen, A. (2015). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4).

[79] Hua, W., Wang, Z., Wang, H., Zheng, K., and Zhou, X. (2016). Understand short texts by harvesting and analyzing semantic knowledge. *IEEE transactions on Knowledge and data Engineering*, 29(3).

[80] Huang, J., Ren, Z., Zhao, W. X., He, G., Wen, J.-R., and Dong, D. (2019). Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 573–581.

[81] Jastrzebski, S., Leśniak, D., and Czarnecki, W. M. (2017). How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks. *arXiv preprint arXiv:1702.02170*.

[82] Javed, F., Hoang, P., Mahoney, T., and McNair, M. (2017). Large-scale occupational skills normalization for online recruitment. In *Twenty-Ninth IAAI Conference*.

[83] Jiang, J. J. and Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*.

[84] Jolliffe, I. T. and Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202.

[85] Joulin, A., Grave, E., Bojanowski, P., and 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: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

[86] Jung, J. J. (2008). Taxonomy alignment for interoperability between heterogeneous virtual organizations. *Expert Systems with Applications*, 34(4):2721–2731.

[87] Jurgens, D., Mohammad, S., Turney, P., and Holyoak, K. (2012). Semeval-2012 task 2: Measuring degrees of relational similarity. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 356–364.

[88] Jurgens, D. and Pilehvar, M. T. (2015). Reserating the awesometastic: An automatic extension of the wordnet taxonomy for novel terms. In *ACL*, pages 1459–1465.

[89] Karp, R. M. (1971). A simple derivation of edmonds' algorithm for optimum branchings. *Networks*, 1(3):265–272.

[90] Khaouja, I., Kassou, I., and Ghogho, M. (2021). A survey on skill identification from online job ads. *IEEE Access*.

[91] Köhler, S., Doelken, S. C., Mungall, C. J., Bauer, S., Firth, H. V., Bailleul-Forestier, I., Black, G. C., Brown, D. L., Brudno, M., Campbell, J., et al. (2014). The human phenotype ontology project: linking molecular biology and disease through phenotype data. *Nucleic acids research*, 42(D1):D966–D974.

[92] Lai, S., Liu, K., He, S., and Zhao, J. (2016). How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14.

[93] Lastra-Díaz, J. J., García-Serrano, A., Batet, M., Fernández, M., and Chirigati, F. (2017). Hesml: A scalable ontology-based semantic similarity measures library with a set of reproducible experiments and a replication dataset. *Information Systems*, 66:97–118.

[94] Leacock, C. and Chodorow, M. (1998). Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2).

[95] Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *NeurIPS*.

[96] Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3.

[97] Lin, D. et al. (1998). An information-theoretic definition of similarity. In *ICML*, volume 98.

[98] Lovaglio, P. G., Cesarini, M., Mercorio, F., and Mezzanzanica, M. (2018). Skills in demand for ICT and statistical occupations: Evidence from web-based job vacancies. *Stat. Anal. Data Min.*, 11(2):78–91.

[99] Luong, M.-T., Socher, R., and Manning, C. D. (2013). Better word representations with recursive neural networks for morphology. In *Proceedings of the seventeenth conference on computational natural language learning*, pages 104–113.

[100] Lv, Z. and Peng, R. (2021). A novel periodic learning ontology matching model based on interactive grasshopper optimization algorithm. *Knowledge-Based Systems*, page 107239.

[101] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *ACL HLT*.

[102] Maedche, A. and Staab, S. (2001). Ontology learning for the semantic web. *IEEE Intelligent systems*, 16(2):72–79.

[103] Mahdisoltani, F., Biega, J., and Suchanek, F. (2014). Yago3: A knowledge base from multilingual wikipedias. In *7th biennial conference on innovative data systems research*. CIDR Conference.

[104] Malandri, L., Mercorio, F., Mezzanzanica, M., and Nobani, N. (2020). Meet: A method for embeddings evaluation for taxonomic data. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 31–38. IEEE.

[105] Malandri, L., Mercorio, F., Mezzanzanica, M., and Nobani, N. (2021). Taxoref: Embeddings evaluation for ai-driven taxonomy refinement. In Oliver, N., Pérez-Cruz, F., Kramer, S., Read, J., and Lozano, J. A., editors, *Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference, ECML PKDD 2021, Bilbao, Spain, September 13-17, 2021, Proceedings, Part III*, volume 12977 of *Lecture Notes in Computer Science*, pages 612–627. Springer.

[106] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[107] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[108] Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751.

[109] Miller, G. A. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1).

[110] Nickel, M. and Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30:6338–6347.

[111] Nickel, M. and Kiela, D. (2018). Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pages 3779–3788. PMLR.

[112] Nickerson, R. C., Varshney, U., and Muntermann, J. (2013). A method for taxonomy development and its application in information systems. *European Journal of Information Systems*, 22(3):336–359.

[113] Ninio, F. (1976). A simple proof of the Perron-Frobenius theorem for positive symmetric matrices. *Journal of Physics A: General Physics*, 9(8):1281–1282.

[114] Otter, D. W., Medina, J. R., and Kalita, J. K. (2020). A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*.

[115] Papoutsoglou, M., Ampatzoglou, A., Mittas, N., and Angelis, L. (2019). Extracting knowledge from on-line sources for software engineering labor market: A mapping study. *IEEE Access*.

[116] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12.

[117] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *EMNLP*.

[118] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. in naacl.

[119] Ponzetto, S. P. and Navigli, R. (2009). Large-scale taxonomy mapping for restructuring and integrating wikipedia. In *IJCAI*, volume 9, pages 2083–2088.

[120] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

[121] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

[123] Real, F. J. Q., Bella, G., McNeill, F., and Bundy, A. (2020). Using domain lexicon and grammar for ontology matching. In *OM@ ISWC*, pages 1–12.

[124] Resnik, P. (1999). Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *JAIR*, 11.

[125] Ristoski, P. and Paulheim, H. (2016). Rdf2vec: Rdf graph embeddings for data mining. In *ISWC*, pages 498–514. Springer.

[126] Rous, B. (2012). Major update to acm's computing classification system. *Commun. ACM*, 55(11):12.

[127] Roy, A. and Pan, S. (2021). Incorporating extra knowledge to enhance word embedding. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4929–4935.

[128] Rubenstein, H. and Goodenough, J. B. (1965). Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

[129] Schlichtkrull, M. and Alonso, H. M. (2016). Msejrku at semeval-2016 task 14: Taxonomy enrichment by evidence ranking. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1337–1341.

[130] Schnabel, T., Labutov, I., Mimno, D., and Joachims, T. (2015). Evaluation methods for unsupervised word embeddings. In *EMNLP*.

[131] Schönbrodt, F. D. and Perugini, M. (2013). At what sample size do correlations stabilize? *Journal of Research in Personality*, 47(5).

[132] Seco, N., Veale, T., and Hayes, J. (2004). An intrinsic information content metric for semantic similarity in wordnet. In *Ecai*, volume 16.

[133] Shang, C., Dash, S., Chowdhury, M. F. M., Mihindukulasooriya, N., and Gliozzo, A. (2020). Taxonomy construction of unseen domains via graph-based cross-domain knowledge transfer. In *Proceedings of the 58th annual meeting of the Association for Computational Linguistics*, pages 2198–2208.

[134] Shen, J., Shen, Z., Xiong, C., Wang, C., Wang, K., and Han, J. (2020). Taxoexpan: Self-supervised taxonomy expansion with position-enhanced graph neural network. In *WWW*, pages 486–497.

[135] Shen, W., Wang, J., Luo, P., and Wang, M. (2012). A graph-based approach for ontology population with named entities. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 345–354.

[136] Singhal, T., Liu, J., Blessing, L. T., and Lim, K. H. (2021). Analyzing scientific publications using domain-specific word embedding and topic modelling. In *2021 IEEE international conference on big data (big data)*, pages 4965–4973. IEEE.

[137] Sumida, A. and Torisawa, K. (2008). Hacking wikipedia for hyponymy relation acquisition. In *IJCNLP*.

[138] Toral, A. and Monachini, M. (2008). Named entity wordnet. In *LREC*.

[139] Torregrossa, F., Allesiardo, R., Claveau, V., Kooli, N., and Gravier, G. (2021). A survey on training and evaluation of word embeddings. *International Journal of Data Science and Analytics*, 11:85–103.

[140] Torregrossa, F., Claveau, V., Kooli, N., Gravier, G., and Allesiardo, R. (2020). On the correlation of word embedding evaluation metrics. In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 4789–4797.

[141] Tsvetkov, Y., Faruqui, M., Ling, W., Lample, G., and Dyer, C. (2015). Evaluation of word vector representations by subspace alignment. In *EMNLP*.

[142] Turrell, A., Speigner, B., Djumalieva, J., Copple, D., and Thurgood, J. (2018). Using job vacancies to understand the effects of labour market mismatch on uk output and productivity. *Bank of England Working Paper*.

[143] UK Commission for Employment and Skills (2015). The importance of LMI, available at https://goo.gl/TtRwvS.

[144] van Dinter, R., Catal, C., and Tekinerdogan, B. (2021). A multi-channel convolutional neural network approach to automate the citation screening process. *Applied Soft Computing*, 112:107765.

[145] Vedula, N., Nicholson, P. K., Ajwani, D., Dutta, S., Sala, A., and Parthasarathy, S. (2018). Enriching taxonomies with functional domain knowledge. In *ACM SIGIR*, pages 745–754.

[146] Velardi, P., Faralli, S., and Navigli, R. (2013). Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707.

[147] Vinel, M., Ryazanov, I., Botov, D., and Nikolaev, I. (2019). Experimental comparison of unsupervised approaches in the task of separating specializations within professions in job vacancies. In *Conference on Artificial Intelligence and Natural Language*, pages 99–112. Springer.

[148] Wang, B., Wang, A., Chen, F., Wang, Y., and Kuo, C.-C. J. (2019). Evaluating word embedding models: Methods and experimental results. *APSIPA transactions on signal and information processing*, 8.

[149] Wang, C., He, X., and Zhou, A. (2017). A short survey on taxonomy learning from text corpora: Issues, resources and recent advances. In *EMLP*, pages 1190–1203.

[150] Wang, J., Kang, C., Chang, Y., and Han, J. (2014). A hierarchical dirichlet model for taxonomy expansion for search engines. In *WWW*, pages 961–970.

[151] Wu, T., Qi, G., Wang, H., Xu, K., and Cui, X. (2016). Cross-lingual taxonomy alignment with bilingual biterm topic model. In *AAAI*, pages 287–293.

[152] Wu, W., Li, H., Wang, H., and Zhu, K. Q. (2012). Probase: A probabilistic taxonomy for text understanding. In *ACM SIGMOD*.

[153] Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection. In *ACL*.

[154] Wu, Z., Zhu, H., Li, G., Cui, Z., Huang, H., Li, J., Chen, E., and Xu, G. (2017). An efficient wikipedia semantic matching approach to text document classification. *Information Sciences*, 393:15–28.

[155] Xing, F., Malandri, L., Zhang, Y., and Cambria, E. (2020). Financial sentiment analysis: An investigation into common mistakes and silver bullets. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 978–987.

[156] Xu, T., Zhu, H., Zhu, C., Li, P., and Xiong, H. (2018). Measuring the popularity of job skills in recruitment market: A multi-criteria approach. In *AAAI*.

[157] Yang, D. and Powers, D. (2006). Verb similarity on the taxonomy of wordnet. In *The Third International WordNet Conference: GWC 2006*. Masaryk University.

[158] Yang, S., Zou, L., Wang, Z., Yan, J., and Wen, J.-R. (2017). Efficiently answering technical questions—a knowledge graph approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31 (1).

[159] Zhang, M., van der Goot, R., and Plank, B. (2023). Escoxlm-r: Multilingual taxonomy-driven pre-training for the job market domain. *arXiv preprint arXiv:2305.12092*.

[160] Zhang, Y., Ahmed, A., Josifovski, V., and Smola, A. (2014). Taxonomy discovery for personalized recommendation. In *Proceedings of the 7th ACM international conference on Web search and data mining*.