Indra: A Word Embedding and Semantic Relatedness Server

Juliano Efson Sales[†], Leonardo Souza[†], Siamak Barzegar[§], Brian Davis[§∓], André Freitas[δ] and Siegfried Handschuh[†]

†Department of Computer Science and Mathematics - University of Passau, Germany
[§]Insight Centre for Data Analytics - National University of Ireland, Galway
[∓]Department of Computer Science - Maynooth University, Ireland
δSchool of Computer Science - The University of Manchester, United Kingdom
{juliano-sales, siegfried.handschuh}@uni-passau.de, siamak.barzegar@insight-centre.org
brian.davis@.mu.ie, andre.freitas@manchester.ac.uk

Abstract

In recent years word embedding/distributional semantic models evolved to become a fundamental component in many natural language processing (NLP) architectures due to their ability of capturing and quantifying semantic associations at scale. Word embedding models can be used to satisfy recurrent tasks in NLP such as lexical and semantic generalisation in machine learning tasks, finding similar or related words and computing semantic relatedness of terms. However, building and consuming specific word embedding models require the setting of a large set of configurations, such as corpus-dependant parameters, distance measures as well as compositional models. Despite their increasing relevance as a component in NLP architectures, existing frameworks provide limited options in their ability to systematically build, parametrise, compare and evaluate different models. To answer this demand, this paper describes INDRA, a multi-lingual word embedding/distributional semantics framework which supports the creation, use and evaluation of word embedding models. In addition to the tool, INDRA also shares more than 65 pre-computed models in 14 languages.

Keywords: word embedding server, semantic relatedness server, semantic toolkit, corpus pre-processor

1. Introduction

Word embedding is a popular semantic model which represents words and sentences in computational linguistics systems and machine learning models. In recent years a large set of algorithms for both generating and consuming word embedding models (WEMs) have been proposed, which includes corpus pre-processing strategies, WEM algorithms or weighting schemes, vector compositions and distance measures (Turney and Pantel, 2010; Lapesa and Evert, 2014; Mitchell and Lapata, 2010). Determining the optimal set of strategies for a given problem demands the support of a tool that facilitates the exploration of the configuration space of parameters.

Furthermore, given the applicability and maturity achieved by these systems and models, they have been promoted from academic prototypes to industry-level applications (Loebbecke and Picot, 2015; Hengstler et al., 2016; Moro et al., 2015). In this new production scenario, a candidate tool should be able to scale to large number of requests and to the construction of models from large corpora, making use of parallel execution and traceability. From the functional point of view, *integrated corpus pre-processing*, *generation of predictive-based and count-based models* and *unified access as a service* are key features.

To support this demand, this paper describes INDRA, a word embedding/distributional semantics framework which supports the creation, use and evaluation of word embedding models. INDRA provides a software infrastructure to facilitate the experimentation and customisation of multilingual WEMs, allowing end-users and applications to consume and operate over multiple word embedding spaces as a service or library.

INDRA is available from two repositories (`github.com/Lambda-3/Indra` and `github.com/Lambda-3/IndraIndexer`) both licensed as open-source software. Additionally, INDRA also provides a Python client (`pyindra`) available via `pip` and from `github.com/Lambda-3/pyindra`.

2. Related Work

S-SPACE is a library to support the construction of count-based distributional methods unifying different approaches in a common JAVA API (Jurgens and Stevens, 2010). DEEPLEARNING4J[1], on the other hand, is a library which concentrates predictive-based models. DEEPLEARNING4J is also written in JAVA and its API contains methods to access word vectors and to find nearest neighbours (kNN).

GENSIM is one of the most popular word-embedding toolkits, mainly credited to its efficient implementation of nearest neighbours function (Řehůřek and Sojka, 2010). GENSIM is written in PYTHON and apart from its kNN function, it supports the generation of predictive-based models and methods to access word vectors.

Following a different motivation, DISSECT (DIStributional SEmantics Composition Tookit) focuses on vector compositions (Dinu et al., 2013). DISSECT is a PYTHON library containing methods to generate vector representation of sentences from the vector of its constituting words. DISSECT partially supports the generation of count-based models and brings an integrated baseline framework for evaluation purposes.

JOBIMTEXT is a semantic similarity tool that implements its own algorithm named JoBim (Biemann et al., 2013). The tool supports the construction of the JoBim model and

[1] https://deeplearning4j.org/

Features	INDRA	GenSim	DeepLearning4J	S-Space	JoBim	DISSECT
Word Embeddings						
Simple vectors	✓	✓	✓	✓		✓
Composed vectors	✓					✓
Translation-based vectors	✓					
Word embeddings as a service	✓					
Semantic Relatedness						
Word pair relatedness	✓	✓	✓	✓	✓	✓
Top-k nearest neighbors	✓	✓	✓	✓	✓	
Score-based nearest neighbors	✓				✓	
Translation-based relatedness	✓					
Composed-vector relatedness	✓					✓
Multiple score functions	✓					✓
Relatedness as a service	✓				✓	
Model Generation and Other Functions						
Integrated corpus pre-processing	✓					
Support to model generation	P/C	P	P	C	JoBim	C*
Support to sparse vector models	✓			✓	✓	✓
Bult-in word desambiguation					✓	
English pre-computed models	✓	✓			✓	
Multi-lingual pre-computed models	✓					
Multi-model querying	✓					

Table 1: List of functionalities and framework coverage. In the line *Support to model generation*, P stands for *predictive-based models* and C for *count-based models*. *DISSECT partially supports the generation of count-based models.

also calculates semantic relatedness of pairs of terms, finds nearest neighbours and offers a native web server.

EasyESA (Carvalho et al., 2014) and DInfra (Barzegar et al., 2015) are also two initiatives to deliver distributional semantics capabilities under a more specific set of distributional semantic models.

From the evaluation point of view, Barzegar et al. (2018c) defined a multi-lingual dataset to measure semantic relatedness in eleven languages.

Table 1 summarises a comparative analysis of the main frameworks and their features. Apart from INDRA, none of the listed frameworks gives support to corpora pre-processing (which will be detailed in Section 3.1.2.). Other limitations addressed by Indra are *(i)* the generation of both count-based and predictive-based models, *(ii)* the support for vector composition and *(iii)* the support for translation-based models.

Finally those libraries offer a limited set of pre-computed models, which makes the process of exploration time-consuming and computationally costly. INDRA aims at covering these gaps by providing an end-to-end infrastructure to build, consume and evaluate multi-lingual word embedding models.

3. Implementation Design

The INDRA PROJECT is divided into two major modules: INDRAINDEXER and INDRA. INDRAINDEXER is responsible for the generation of the models, whereas INDRA implements the consumption methods.

INDRA is designed to be a stand-alone library and also a web service. Figure 1 depicts the main components of its architecture. INDRAINDEXER supports the generation of WEMs directly from text files (Wikipedia-dump or plain-text formats), passing through the corpus pre-processing and multiword expression identification, to the model generation itself. INDRA dynamically builds the pipeline based on the metadata information produced during the model generation. This strategy guarantees that the same set of pre-processing operations are consistently applied to the input query. Additionally, the translation-based word embedding (Freitas et al., 2016; Barzegar et al., 2018b) can be conveniently activated in the pipeline as described in Section 4..

Different languages, domains and application scenarios require different parametrisations of the underlying embedding models. Together with the availability of pre-generated models, INDRA's system architecture favours the exploration of a large grid of parameters. INDRA currently shares more than 65 pre-computed models which varies in languages, model algorithms and corpora (general-purpose and domain-specific). The list of available models are in the Github project's Wiki.

3.1. Functionalities

Table 1 shows the functionalities implemented in INDRA, among which the following set deserves our attention: *text pre-processing*, *model generation*, *semantic relatedness*, *nearest neighbours*, *vector server*, *semantic relatedness*, *vector compositions* and the *support to translation-based models*.

3.1.1. Text Pre-processing

One important step in the construction of word embeddings models is pre-processing the texts. Defining the tokenisation strategy, which depends on the language, whether or

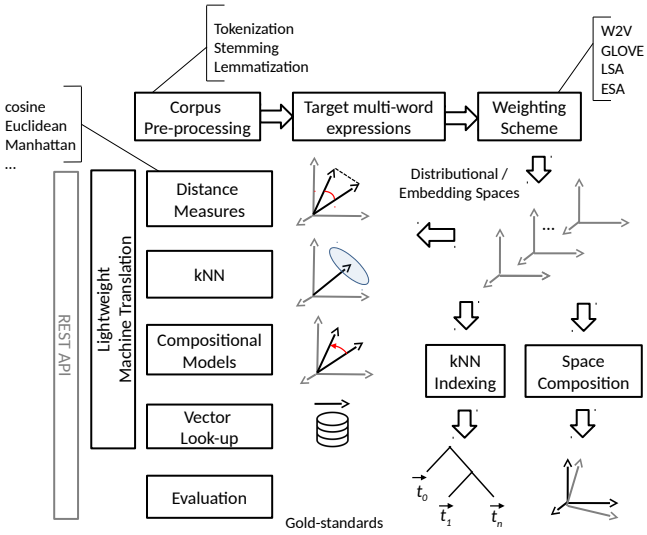

Figure 1: High-level architecture of the Indra framework.

not words must be lower-cased or stemmed is part of the pre-processing step. Furthermore, the pre-process strategy should be stated consistently during both construction and consumption phases as exemplified further.

The corpus pre-processor is responsible for defining the tokenisation strategy and the tokens' subsequent transformations. It defines, for example, if `United States of America` corresponds to a unique or to multiple tokens. *Stem* and *lowercase* are two other popular transformations also supported by the pre-processor.

INDRA uses the Lucene's `StandardTokenizer`, which implements the Unicode Text Segmentation algorithm based on the Word Break rules defined in the Unicode Standard Annex #29 (Davis and Iancu, 2017). Additionally, INDRA allows users to specify a customised list of multi-word expressions which will be considered a unique token, independently of the tokeniser rules. This mechanism allows, for example, modelling a unique vector for named entities such as *Nelson Mandela* and *Republic of Austria*.

As in the context of WEM numbers are usually disregarded tokens, the pre-processing step allows replacing them by a default placeholder (¡NUMBER¿). INDRA pre-processor also allows specifying stopwords, whose occurrences are removed from the text. Table 2 shows the full list of operations supported by the pre-processor.

The pre-processor is defined as a package that is attached to both INDRAINDEXER and INDRA in order to guarantee that the consuming functions apply the same set of operations in retrieval time.

3.1.2. Model Generation

INDRAINDEXER is the module responsible for the generation of word embedding models. It defines a unified interface to generate predictive-based models (e.g. Skip-gram (Mikolov et al., 2013) and Global Vectors (Pennington et al., 2014)) and count-based models (e.g. Latent Semantic Analysis (Dumais et al., 1988) and Explicit Semantic Analysis (Gabrilovich and Markovitch, 2007)) whose implementation comes from the libraries DeepLearning4J[2] and S-Space (Jurgens and Stevens, 2010) respectively. In addition to creating a unified interface for WEM algorithms, INDRAINDEXER integrates the corpus pre-processor package.

INDRAINDEXER receives as input the pre-processed corpus and outputs the vectors in binary files in a format compatible with GenSim (Řehůřek and Sojka, 2010). In addition to the vector file, INDRAINDEXER also generates a metadata file containing all the parameters from both the pre-processing and generation steps. Figure 3 shows an example of a metadata file.

```
{
  {
    "windowSize" : 5,
    "minWordFrequency" : 5,
    "corpusMetadata" : {
       "corpusName" : "wiki-2014",
       "stopWords" : ["been", "don't", ...],
       "replaceNumbers" : false,
       "applyStemmer" : 3,
       "removeAccents" : true,
       "maxTokenLength" : 100,
       "minTokenLength" : 3,
       "description" : null,
       "language" : "en",
       "encoding" : "UTF-8",
       "applyLowercase" : true
    },
    "vocabSize" : 1181258,
    "sparse"  :false,
    "model" : "w2v",
    "dimensions" : 300
}
}
```

Figure 2: Example of metadata file generated by INDRAINDEXER which describes how the user configured both the pre-processor and the WEM generator.

During the consumption phase, INDRA applies the same set of options to guarantee consistence. For instance, let's assume a given model was generated by applying the stemmer and lowercase to the tokens. It means that the term `University` is represented in the model as `univers`. When it is required to retrieve the vector representation of `University`, INDRA guarantees this consistence by executing the pre-processing steps in the query at runtime. This method simplifies the execution of experiments that consumes models using different set of pre-processing transformations.

3.1.3. Nearest Neighbours

Given a term and an integer k, the Nearest Neighbours function lists the set of its k closest terms. This method is applied, for instance, in *topic modelling* (Řehůřek and Sojka, 2010) and *vocabulary expansion* (Atzori et al., 2018). INDRA implements this function using the SPOTIFY ANNOY

[2]https://deeplearning4j.org/

Parameter	Description/Options
input format	Wikipedia-dump format or plain texts from one or multiple files.
language	14 supported languages.
set of stopwords	a set of tokens to be removed.
set of multi-word expressions	set of sequences of tokens that should be considered a unique token.
apply lowercase	lowercase the tokens.
apply stemmer	applies the Poter Stemmer in the tokens.
remove accents	remove the accents of words.
replace numbers	replaces numbers for the place holder $<NUMBER>$.
min	set a minimum acceptable token size.
max	set a maximum acceptable token size.

Table 2: Parameters supported by the INDRA's pre-processing package.

library [3], since a preliminary study suggests ANNOY's performance is an order of magnitude better performing than GENSIM's (Řehůřek, 2014). In addition to the identification of term's neighbours, the function also accepts a vector as input.

Another related function present in INDRA is the selection based on thresholds, in which INDRA gets a query term and a set of target terms as inputs, and returns those target terms whose relatedness score is greater (or lower) than a given threshold. The threshold can be determined both statically or dynamically(Freitas et al., 2012).

3.1.4. Vector Server

As a primary use, INDRA acts as a central repository of WEMs, serving vectors for terms in different languages and models. The set of pre-processed models allows the user to experiment different WEMs configurations as a one-stop-shop fashion. INDRA can act as a central server in an enterprise context, or as a local library in more constrained environments.

3.1.5. Semantic Relatedness

Natural language understanding systems use semantic relatedness in fine-grained tasks such as *word disambiguation* (Freitas et al., 2013) or more coarse-grained such as *paraphrase detection* (Sales et al., 2016; Silva et al., 2018), *semantic parsing* (Sales et al., 2018) and *question answering* (Freitas, 2015). INDRA implements two semantic relatedness methods. The first is the pair-wise semantic relatedness in which the user provides pairs of terms to calculate their semantic relatedness. The other option is integrated to the nearest neighbours function which returns the relatedness of the *k* closest terms.

Additionally Indra can support the application of various distance or correlation measures (Lapesa and Evert, 2014). Currently INDRA supports more than ten different distance and correlation functions, including *Cosine*, *Jaccard*, *Euclidean* and *Spearman Correlation*.

3.1.6. Vector Compositions

In simple terms, *vector composition* aims at generating a vector representation of phrases and sentences from the combination of individual vectors of its compound terms(Kartsaklis, 2014). For example, the vector representation of `modern Democratic Party` is generated by the composition of the corresponding vectors of the three compounding terms `modern`, `Democratic` and `Party`. Currently, INDRA implements three composition methods (*Sum*, *Normalised Sum* and *Average*) and supports the extension of user-defined functions as described in Section 3.2..

Vector composition is automatically associated to the semantic relatedness function or the retrieval of vectors. Whenever a expression comprehending more than one token is submitted, INDRA composes their corresponding vectors before executing the required function.

3.1.7. Support to Translation-based Models

Some languages do not have large text corpora publicly available. As word embedding models are sensitive to the corpus size, (Freitas et al., 2016; Barzegar et al., 2018a) propose the use of *translation-based models*. In simple words, the translation-based strategy translates the original query terms to a second language for which a high quality WEM is available. INDRA gives native support to this operation as described in Section 4..

3.2. Extensibility

INDRA implements a plugin-based extensible mechanism built on the top of the JAVA SERVICE API which allows including new *compositional methods*, *score functions* and *threshold functions* without recompiling Indra's code. To do so, it is required to pack the new functions' implementations in a JAR file and place it in the INDRA's *classpath*[4].

4. Use Examples

INDRA's service exposes the functions as POST methods, whose data are passed as a JSON payload. For simplicity, we suppress the request headers to concentrate our attention in the payload itself.

Every request has at least three mandatory fields: *language*, *model* and *corpus*. The first naturally specifies the request's language. The second and the third name respectively the algorithm and the corpus from which the word embedding model were generated. This trio is the model's unique identifier.

[3] https://github.com/spotify/annoy

[4] For more information about The Java Archive (JAR) and CLASSPATH, please refer to official Java documentation.

Word Embedding: Figure 3 shows a payload to the endpoint `/vectors` which returns the respective word embedding vectors of the `terms`. In the case a term is composed of more than one token, `termComposition` is applied.

```
{
  "corpus": "googlenews300neg",
  "model": "W2V",
  "language": "EN",
  "terms": ["love", "best of you"],
  "termComposition" : "AVERAGE"
}
```

Figure 3: Payload to request the word embedding of the term *love* and the expression *best of you*.

Pair Relatedness: The endpoint `/relatedness` returns the semantic relatedness of the `pairs`. The relatedness operation is defined by the field `scoreFunction` as shown in Figure 4. In the case that `termComposition` is not defined, the default function is used.

```
{
    "corpus": "wiki-2014",
    "model": "ESA",
    "language": "PT",
    "scoreFunction": "COSINE",
    "pairs": [{
        "t1": "economia",
        "t2": "Rio de Janeiro"
        },
        {
        "t1": "economia",
        "t2": "soja"
    }]
}
```

Figure 4: Payload to request the cosine relatedness of two pairs of terms in Portuguese.

One-to-many Relatedness: The endpoint `/relatedness/otm` returns the semantic relatedness of `one` term against a set of `many` terms. Similarly to the previous operation, the relatedness operation is defined by the field `scoreFunction` as shown in Figure 5.

Nearest Neighbours: The nearest neighbours function is exposed in two methods. The endpoint `/neighbors/relatedness` returns the relatedness score between the target terms and theirs top-k neighbours, according to the payload depicted in Figure 6.
When submitting the same payload to the endpoint `/neighbors/vectors`, the service returns the list of the neighbours and their respective vectors.

Translated-based Word Embbedings: The requests support the translated-based function, in which the vectors is extracted from the corresponding English corpus after translating the terms from the original query. The translated-based function is activated by appending `"mt"=true` in the payload. INDRA offers seven pre-computed light-weight translation models.
For a complete description of the methods and parameters, please refer to the project documentation.

```
{
    "corpus": "wiki-2014",
    "model": "ESA",
    "language": "EN",
    "scoreFunction": "JACCARD",
    "one": "Germany",
    "many" : ["France", "China", "Brazil"]
}
```

Figure 5: Payload to request the Jaccard relatedness of the implicit pairs [Germany, France], [Germany, China] and [Germany, Brazil].

```
{
  "corpus": "wiki-2014",
  "model": "GLOVE",
  "language": "SV",
  "topk": 10,
  "terms": ["ekonomi", "flicka", "frihet"]
}
```

Figure 6: Payload to request the 10 most related terms individually to *ekonomi*, *flicka* and *frihet*. This call returns three set of 10 terms, each one corresponding to one of the terms.

5. Python Client

Our project also offers a client to access the service from Python application. The `pyindra` package is available in the `pip` repository.
The client source code is at `https://github.com/Lambda-3/pyindra`.

6. Summary

Many applications of word embedding models require the customisation of the models in the direction of domain-specific vocabularies, specific languages or specific semantic approximation behaviour (e.g. paradigmatic vs syntagmatic behaviour), distance measures as well as compositional models. This work introduces the INDRA framework which manages the complexity of experimenting and using word embedding models in exploratory scenarios and production environments. INDRA also shares more than 65 pre-computed models and is available as an open-source software.

7. Acknowledgment

This publication has emanated from research partially supported by the National Council for Scientific and Technological Development, Brazil (CNPq).

8. References

Atzori, M., Balloccu, S., and Bellanti, A. (2018). Unsupervised singleton expansion from free tex. In *2018 IEEE*

Twelfth International Conference on Semantic Computing (ICSC), Jan.

Barzegar, S., Sales, J. E., Freitas, A., Handschuh, S., and Davis, B. (2015). Dinfra: A one stop shop for computing multilingual semantic relatedness. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 1027–1028, New York, NY, USA. ACM.

Barzegar, S., Davis, B., Handschuh, S., , and Freitas, A. (2018a). Multilingual semantic relatedness using lightweight machine translation. In *2018 IEEE Twelfth International Conference on Semantic Computing (ICSC)*, Jan.

Barzegar, S., Davis, B., Handschuh, S., and Freitas, A. (2018b). Multilingual semantic relatedness using lightweight machine translation. In *IEEE International Conference on Semantic Computing*. IEEE.

Barzegar, S., Davis, B., Zarrouk, M., Handschuh, S., and Freitas, A. a. (2018c). Semr-11: A multi-lingual gold-standard for semantic similarity and relatedness for eleven languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May. European Language Resources Association (ELRA).

Biemann, C., Riedl, and M. (2013). Text: Now in 2d! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95, April.

Carvalho, D. S., Calli, C., Freitas, A., and Curry, E. (2014). Easyesa: A low-effort infrastructure for explicit semantic analysis. In *Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014.*, pages 177–180.

Davis, M. and Iancu, L. (2017). Unicode text segmentation. Technical report, The Unicode Consortium, June.

Dinu, G., Pham, N. T., and Baroni, M. (2013). Dissect - distributional semantics composition toolkit. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 31–36, Sofia, Bulgaria, August. Association for Computational Linguistics.

Dumais, S. T., Furnas, G. W., Landauer, T. K., Deerwester, S., and Harshman, R. (1988). Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, pages 281–285, New York, NY, USA. ACM.

Freitas, A., Curry, E., and O'Riain, S. (2012). Distributional approach for terminological semantic search on the linked data web. In *27th ACM Symposium On Applied Computing (SAC), Semantic Web and Applications (SWA)*, Sept.

Freitas, A., Oliveira, J. G., O'Riain, S., da Silva, J. C. P., and Curry, E. (2013). Querying linked data graphs using semantic relatedness: A vocabulary independent approach. *Data Knowl. Eng.*, 88:126–141.

Freitas, A., Barzegar, S., Sales, J. E., Handschuh, S., and Davis, B., (2016). *Semantic Relatedness for All (Languages): A Comparative Analysis of Multilingual Semantic Relatedness Using Machine Translation*, pages 212–222. Springer International Publishing, Cham.

Freitas, A. (2015). *Schema-agnositc queries over large-schema databases: a distributional semantics approach.* Ph.D. thesis, Digital Enterprise Research Institute (DERI), National University of Ireland, Galway.

Gabrilovich, E. and Markovitch, S. (2007). Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Hengstler, M., Enkel, E., and Duelli, S. (2016). Applied artificial intelligence and trust - the case of autonomous vehicles and medical assistance devices. *Technological Forecasting and Social Change*, 105(Supplement C):105 – 120.

Jurgens, D. and Stevens, K. (2010). The s-space package: An open source package for word space models. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 30–35, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kartsaklis, D. (2014). Compositional operators in distributional semantics. *Springer Science Reviews*, 2(1):161–177, Dec.

Lapesa, G. and Evert, S. (2014). A large scale evaluation of distributional semantic models: Parameters, interactions and model selection. *Transactions of the Association for Computational Linguistics*, 2:531–545.

Loebbecke, C. and Picot, A. (2015). Reflections on societal and business model transformation arising from digitization and big data analytics: A research agenda. *The Journal of Strategic Information Systems*, 24(3):149 – 157.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *ICLR Workshop*.

Mitchell, J. and Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Moro, S., Cortez, P., and Rita, P. (2015). Business intelligence in banking: A literature analysis from 2002 to 2013 using text mining and latent dirichlet allocation. *Expert Systems with Applications*, 42(3):1314 – 1324.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Řehůřek, R. (2014). Performance shootout of nearest neighbours: Querying. https://rare-technologies.com/performance-shootout-of-nearest-neighbours-querying/.

Sales, J. E., Freitas, A., Davis, B., and Handschuh, S.

(2016). A compositional-distributional semantic model for searching complex entity categories. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 199–208, Berlin, Germany, August. Association for Computational Linguistics.

Sales, J. E., Freitas, A., and Handschuh, S. (2018). An open vocabulary semantic parser for end-user programming using natural language. In *2018 IEEE Twelfth International Conference on Semantic Computing (ICSC)*, Jan.

Silva, V., Handschuh, S., and Freitas, A., (2018). *Recognizing and Justifying Text Entailment through Distributional Navigation on Definition Graphs*. AAAI.

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, January.