UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Embedding Propagation over Heterogeneous Information Networks

Paulo Ricardo Viviurka do Carmo

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Paulo Ricardo Viviurka do Carmo

Embedding Propagation over Heterogeneous Information Networks

Dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Ricardo Marcondes Marcacini

USP – São Carlos
December 2022

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Viviurka do Carmo, Paulo Ricardo
V858e Embedding Propagation over Heterogeneous Information Networks / Paulo Ricardo Viviurka do Carmo; orientador Ricardo Marcondes Marcacini. -- São Carlos, 2022.
103 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2022.

1. Embedding propagation. 2. Network embedding. 3. Heterogeneous information network. I. Marcondes Marcacini, Ricardo, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Paulo Ricardo Viviurka do Carmo

Propagação de *Embeddings* em Redes Heterogêneas de Informação

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Ricardo Marcondes Marcacini

USP – São Carlos
Dezembro de 2022

Dedico este trabalho a todas as pessoas que me quiseram bem,
principalmente àqueles que me apoiaram em algum momento da minha vida.

ACKNOWLEDGEMENTS

Agradeço primeiramente aos meus pais Roberto Aparecido do Carmo e Lucia Viviurka do Carmo por me apoiarem e me proverem o privilégio de continuar os meus estudos;

Ao meu irmão Danilo Augusto Viviurka do Carmo por, ao crescer comigo, me ajudar a me tornar quem eu sou.

Ao meu orientador Dr. Ricardo Marcondes Marcacini por todo o apoio durante o mestrado, me orientando e me ajudando com tudo o que precisei durante o mestrado, sempre prestativo e atencioso.

Aos meus colegas de laboratório Ivan Filho e ngelo Mendes pelas colaborações e parceria nas disciplinas durante o mestrado.

Aos amigos Renan, André, Alex, Gabriel, João e Mariana que me ajudaram em algum momento, seja me escutando falar animado do meu projeto ou desabafando.

Ao meu antigo orientador Sandro Rautenberg que me ensinou a apreciar a pesquisa e me ajudou a iniciar essa jornada.

Aos meus supervisores do estágio internacional Edgard Marx e Prof. Dr. Thomas Riechert por me proporcionarem essa oportunidade de pesquisar em um projeto interessante e de conhecer um outro ambiente de pesquisa e me imergir em outra cultura.

A Capes (processo número 88887.513429/2020-00) pelo apoio financeiro, me proporcionando ter dedicação exclusiva ao projeto de mestrado.

“Nós vivemos em uma ilha cercada por um mar de ignorância.
Conforme nosso conhecimento cresce, o mesmo acontece com a costa da nossa ignorância.”
(John Archibald Wheeler)

ABSTRACT

DO CARMO, P. R. V. **Embedding Propagation over Heterogeneous Information Networks**. 2022. 100 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2022.

In order to use text data in machine learning tasks, they must be cleaned and transformed to a structured representation. Recently, neural embeddings have been used to encode text data in low dimensionality latent spaces. For example, BERT pre-trained neural language models can position words, sentences, or documents with fixed dimension embedding vectors. Another way to model text data is to use heterogeneous information networks. That structure models multi-typed data respecting relations and characteristics. Heterogeneous information networks also have their challenges for use with off-the-shelf machine learning methods. Network embedding methods allow the extraction of embedding vectors for each node in an information network. However, these methods usually use only network topology, and sometimes, metadata for the relationships. Embedding propagation methods allow previously generated features with pre-trained methods to be propagated through all network nodes. Information networks that contain some nodes with textual information can use pre-trained neural language models features for propagation. This master's dissertation presents an embedding propagation method for heterogeneous information networks with some textual nodes. The proposed method combines pre-trained neural language models to the topology of heterogeneous information networks through a regularization function to generate embedding for non-textual nodes. Three papers on use case experiments to evaluate and validate the proposed method are presented, where one paper extends the experiments from another: (1) *Embedding Propagation over Heterogeneous Event Networks* presents the results of the proposed method for event analysis where it achieved the best performance by at least 3% *MRR@k* in all scenarios; (2) *TRENCHANT: TRENd prediCtion on Heterogeneous informAtion NeTworks* extends *Commodities trend link prediction on heterogeneous information networks* where the proposed method is evaluated against network embeddings in the task of predicting price trends for commodities, and it achieved the best performance in some scenarios, where its best results 8% better $F1$ when predicting weekly soybean price trends; and (3) *NatUKE: Benchmark for Natural Product Knowledge Extraction from Academic Literature* that evaluates the use of network embedding methods for unsupervised knowledge extraction and the proposed method achieved the best performance in most scenarios, more notably it achieved 43% more $Hits@1$ than baselines when extraction the isolation process type to obtain a molecule from a certain species. The presented papers show, in three different use cases and experiments, that the proposed method achieves the research goals of propagating the initial embedding from some textual nodes to the remaining nodes in a heterogeneous information network and allowing dynamic insertion of new nodes in the embedding propagation process.

Keywords: Embedding propagation, Network embedding, Heterogeneous information network.

RESUMO

DO CARMO, P. R. V. **Propagação de *Embeddings* em Redes Heterogêneas de Informação**. 2022. 100 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2022.

Dados textuais precisam ser limpos e transformados para representações estruturadas antes de serem utilizados em cenários de aprendizado de máquina. Recentemente, *embeddings* estão sendo utilizadas para representar dados textuais. Por exemplo, o modelo de linguagem neurais pré-treinado BERT podem posicionar palavras, sentenças ou textos em *embeddings* dentro de um espaço vetorial de dimensão fixa. Outra forma de modelar dados textuais é a utilização de redes heterogêneas de informação. Essa estrutura permite a modelagem de relações complexas por meio de nós e conexões de dados textuais de diferentes domínios com conexões explícitas. Por outro lado, redes de informação possuem seus próprios desafios quanto a utilização direta em métodos tradicionais de aprendizado de máquina. Métodos de *network embedding* podem ser utilizados para gerarem *embeddings* de nós com relação a topologia da rede, tipos de relações e até mesmo rótulos. Entretanto esses métodos normalmente exploram apenas a topologia, e em alguns casos, metadados dos relacionamentos em uma rede. Métodos de propagação de *embeddings* foram desenvolvidos com o objetivo de distribuir vetores de características gerados a partir de outros modelos. Para redes de informação que possuem alguns nós com dados textuais modelos de linguagem pré-treinados podem ser propagados respeitando a topologia e outros dados das redes para a geração de uma *embedding* final. Esta dissertação de mestrado apresenta um método de propagação de *embeddings* para redes heterogêneas de informação que representam dados textuais. O método proposto propaga as *embeddings* de nós textuais por toda a rede por meio de uma função de regularização. Três artigos de caso de uso que avaliam e validam o método também são apresentados: (1) *Embedding Propagation over Heterogeneous Event Networks* mostra o desempenho do método proposto para análise de eventos onde sua performance supera a literatura por mais de 3% $MRR@k$ em todos os cenários; (2) *TRENCHANT: TRENd prediCtion on Heterogeneous informAtion NeTworks* que é uma extensão de *Commodities trend link prediction on heterogeneous information networks* onde o método proposto é avaliado em relação a métodos de *network embedding* da literatura na tarefa de predição de preços de *commodities* e atinge performance superior a literatura em alguns cenários, onde obteve 8% melhor $F1$ predizendo *trends* de preços semanais da soja; e (3) *NatUKE: Benchmark for Natural Product Knowledge Extraction from Academic Literature* que avalia a utilização de métodos de *network embedding* para a extração de conhecimento não-supervisionada e o método proposto obteve a melhor performance na maior parte dos cenários, sendo que em sua melhor performance obteve 43% mais $Hits@1$ que a literatura extraindo o tipo de isolamento necessário para obter certa molécula de uma espécia de planta. Esses artigos mostram por meio de experimentos e

resultados que o método proposto, ao utilizar uma função de regularização para a propagação, atinge os objetivos de pesquisa de propagar uma *embedding* inicial de alguns nós com dados textuais para os nós restantes de uma rede heterogênea de informação e permitir a inserção dinâmica de novos nós ao processo de propagação de *embeddings*.

Palavras-chave: Propagação de *embeddings*, *Network embedding*, Redes heterogêneas.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

CHAPTER

1

INTRODUCTION

1.1 Context and initial remarks

Machine accessible text data has been exponentially growing with the development of the internet (REZENDE, 2003; AGGARWAL; ZHAI, 2012). However, text data is not structured, because humans are able to consume information from unstructured data sources, meanwhile machines can't do the same automatically. For it to be used in real-world applications like predicting who is involved with a certain event or extracting meaningful information from technical texts, it must first be cleaned and transformed to a structured representation for machine learning (ML) (AGGARWAL, 2018).

Usually, textual data is processed and mapped into vectors before ML is applied. The most basic technique for processing text is called Bag of Words (BoW), and this representation creates a vector with the count of tokens for each document in a textual dataset (AGGARWAL; ZHAI, 2012). The BoW technique creates sparse vectors with the size of the lexicon, which is inefficient when processing large textual datasets. In order to circumvent this challenge Mikolov *et al.* (2013) proposed a method called Word2Vec that uses ML and the statistical knowledge of embeddings to generate dense vectors that represent words. Even though Word2Vec is a powerful representation method, pre-trained in billions of texts, its embedding vectors can not differentiate words with more than one meaning in different contexts. To deal with this shortcoming Devlin *et al.* (2018) proposed a method that uses bi-directional transformers networks to embed context into the dense vectors.

Another way to model text in ML is to use heterogeneous information networks (HINs) or knowledge graphs (KGs). These data structures allow complex multi-typed data to be organized and stored with its relations. For example, the authors Rossi, Lopes and Rezende (2014) use a HIN that connects each document to their tokens to model a bi-partite HIN. They also allow domain data to be modeled with its relations. For example, events can be extracted from news texts,

and HINs can be used to model their structure, like names of places, people, and organizations as objects from a HIN (SHI *et al.*, 2016; CHEN; LI, 2020). Figure 1 presents a heterogeneous information network of event data extracted from news text.

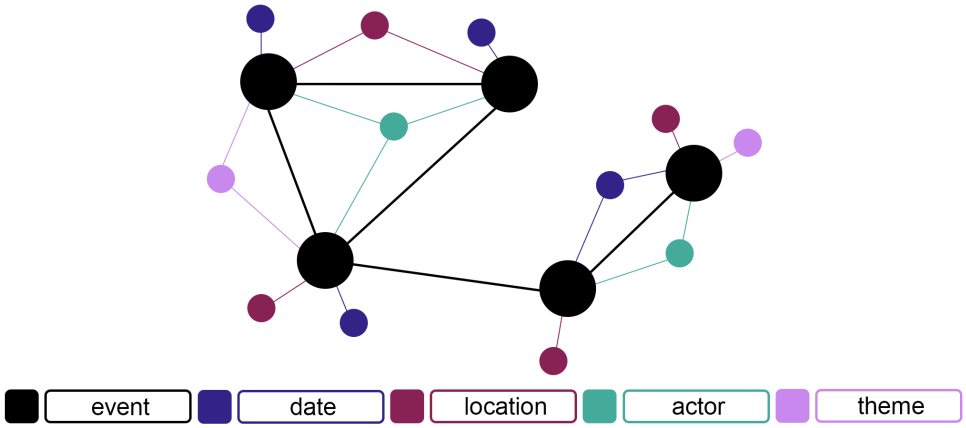


Figure 1 – Representation of a HIN that contains data related to events (event, date, location, actor, and theme). Source: authors.

Although HINs allow different data types to be modeled explicitly, they also impose challenges for use in ML tasks like clustering, classification and others. Network embedding methods circumvent these difficulties by generating embedding vectors that represent the nodes within a fixed dimension vector space while considering topology and even types of relations from HINs (CHANG *et al.*, 2015; HUANG; MAMOULIS, 2017; SETTY; HOSE, 2018; CUI *et al.*, 2018; WU *et al.*, 2020). The embedding vectors then allow off-the-shelf ML methods to be applied when considering HINs nodes directly.

Even though network embedding methods allow the generation of dense vectors, the more complex structure of HINs limits the usage of pre-trained embedding algorithms, which reduces performance on smaller networks. Embedding propagation methods can use pre-existing embedding features available for some nodes in a HIN to generate embeddings for all nodes (DURAN; NIEPERT, 2017; YANG; ZHANG; HAN, 2019; HAMILTON; YING; LESKOVEC, 2017). For textual data neural embedding propagation methods can be used as startup embedding for propagation (MIKOLOV *et al.*, 2013; DEVLIN *et al.*, 2018). This way pre-trained language models can be used to generate features that embed general knowledge from training in massive corpora and maintain a constant vector space that enables dynamic updates within an ML-based application.

This master's dissertation proposes an embedding propagation method for HINs that contain textual data in some objects. More specifically, the embedding propagation method proposed is based on a regularization function that propagates a contextual text embedding from a neural language model called Bidirectional Encoder Representations from Transformers (BERT) (DEVLIN *et al.*, 2018) to all nodes on the HIN. BERT's contextual embeddings provide both contextual positional embeddings and general knowledge from the pre-training. BERT's

architecture also allows for model variations and fine-tunings that enable multilingual processing and even more context recognition.

1.2 Research challenges

Despite its benefits, HINs do not solve all the challenges of processing text data. So, before presenting the research goals of proposing an embedding propagation method it is necessary to understand these challenges. They are summarized below:

- **HINs** complexity and flexibility for modeling data adds complexity to extracting information from the relations and their metadata. This extra complexity brings high computational costs and reduces the number of off-the-shelf methods that can be used. Personalized methods also need much tuning to achieve good performance on different networks.
- **Network embedding** methods are usually tailored for homogeneous information networks, thus ignoring crucial metadata when processing HINs. Network embedding methods also require tuning hyperparameters, mainly when the method aims to extract metadata-like types of relations from heterogeneous information networks. Traditional network embedding methods also ignore text nodes that might hold complementary information. They also do not allow updates to the embeddings on the fly, requiring the entire dataset to be recalculated.

1.3 Research goals

This master's dissertation presents an embedding propagation method that propagates the initial embedding from some textual nodes to the remaining nodes in a HIN and allows the dynamic insertion of new nodes in the embedding propagation process.

More specifically, the embedding propagation process must use initial textual embeddings that will enrich topology embeddings and adapt to them. This method must also maintain a fixed embedding space, allowing for dynamic updates whenever new data is collected and added to the HIN.

The capability of propagation text embeddings to generate network embeddings should allow better performance in comparison to network embedding methods. Also, the dynamic insertion capability must ensure better real world usability by reducing execution times when using the proposed method for inference.

1.4 Main contributions and results

The main contributions of this dissertation are:

- **Embedding Propagation over Heterogeneous Event Networks** (EPHEN) is the introduced embedding propagation method based on a regularization function. The method was published in Carmo and Marcacini (2021) with an evaluation setup compared to other network embedding methods. An evaluation was conducted in an event dataset from GDELT project[1] that compared EPHEN's embedding vectors to other network embedding methods in a link prediction task. The paper also links the source code and the used datasets. In the end, the published paper shows that EPHEN is dynamically updatable thanks to using a BERT model to start the embedding propagation process.

- **Commodities trend link prediction on heterogeneous information networks** is a use case paper published in Carmo, Filho and Marcacini (2021). The paper compares an embedding propagation method called Trend Prediction in Heterogeneous Information Networks (TPHIN) to other network embedding methods in a trend node prediction scenario. Which is the proposed method's name for this paper. The paper shows how embedding propagation can perform well in a real-world scenario. The paper also provides source code and the information networks used in the experiments.

- **TRENCHANT: TRENd prediCtion on Heterogeneous informAtion NeTworks** is an extension of the previous paper, accepted but not yet published in Carmo, Filho and Marcacini (2022). In this paper TPHIN is named TREHNCHANT and its structure is refined and extended with a BERT fine-tuning pipeline. The fine-tuning pipeline proposed uses the similarity between central event nodes within the network to regularize similarity between phrases on the BERT model, adding knowledge from the HIN topology to the language model. The results for fine-tuned TPHIN show different behavior than regular TPHIN, indicating that fine-tuning the BERT model starting the embeddings can improve the performance of the entire embedding propagation model in some cases. This work also provides open source code and uses the same dataset as the previous paper in a different experimental setup.

- **NatUKE: Benchmark for Natural Product Knowledge Extraction from Academic Literature** compares EPHEN from the paper Carmo and Marcacini (2021) to other network embedding methods in an information extraction task. This benchmark work evaluates these methods for information extraction on academic papers and is accepted but not yet published in Carmo *et al.* (2023). All the methods are compared in a dynamically evolving knowledge graph completion scenario. The nodes of interest represent different characteristics of chemical compounds like compound name or bioactivity, and their automatic extraction is desired to update a chemical network in the future. EPHEN achieved the best performance in most tasks meaning that the embedding propagation method can be used for different domains and tasks.

[1] Accessible at: <https://www.gdeltproject.org>.

1.5 Dissertation organization

This master's dissertation was organized as follows:

- Chapter 1 presents the context and motivation of this master's dissertation. Research challenges and goals are also presented to further contextualize the domain and objectives. Main contributions and results are also shown to give further context to the dissertation's text.
- Chapter 2 presents basic concepts involving text mining and information network representation learning. The text mining concepts further explain pre-processing, pattern extraction, and post-processing of machine learning techniques applied to text mining. In information network representation, learning regularization methods and network embedding are explained. This chapter also presents related works in network embeddings, regularization event analysis, and graph completion.
- The proposed method is presented in Chapter 3 together with the knowledge graph completion techniques used to evaluate the method in different tasks.
- Chapter 4 presents the paper where the proposed embedding propagation method is applied to event analysis tasks.
- Chapter 5 presents the extension of a paper that was the initial use case experiment. The proposed embedding propagation method is applied to a trend prediction of commodities prices. These works use event data related to commodities and the time series to their prices.
- Chapter 6 presents the experiment and results of a benchmark of information extraction of chemical compounds from academic data sources using network embedding methods, including the proposed embedding propagation method. This chapter shows that the proposed embedding propagation method is versatile. It also presents details of the three month guest research work in the Hochschule für Technik Wirtschaft und Kultur (HTWK) at Leipzig, Germany.
- Chapter 7 concludes this master's dissertation by discussing the contributions and innovations achieved throughout the research. It also presents publications and collaborations realized during the master's. And finally, it presents some limatations and possible future work for this research.

CHAPTER

2

THEORETICAL FOUNDATION AND RELATED WORKS

This Chapter presents the basic concepts of text mining, highlighting some use cases for extracting different data types. It also presents some HIN representation learning techniques, like regularization and network embedding. At last, it presents some related works using these concepts and methods to position this dissertation within these research areas.

2.1 Text mining

Text data has grown exponentially with news and social networks on the internet (REZENDE, 2003; AGGARWAL; ZHAI, 2012). Thanks to that growth, text mining techniques and methods development have also been increasing. Rezende (2003) describes a text mining organization cycle that allows the process to be implemented in different datasets. This cycle is represented in Figure 2, and the steps can be described as:

- **Domain Knowledge**: it is the first step where the researcher is expected to conduct an initial study of the text's domain within the dataset as well as other sources and related works. The obtained knowledge is then used for defining the objectives and validate the obtained results according to these expectations;
- **Pre-processing**: in this second step, the researcher uses the obtained domain knowledge to implement techniques like text cleaning and normalization as well as to choose a representation model like information networks. This step is further explained in the subsection 2.1.1;
- **Pattern extraction**: in the third step, the researcher applies methods like statistical analysis and machine learning to extract patterns within the dataset. This step is further explained in the subsection 2.1.2;

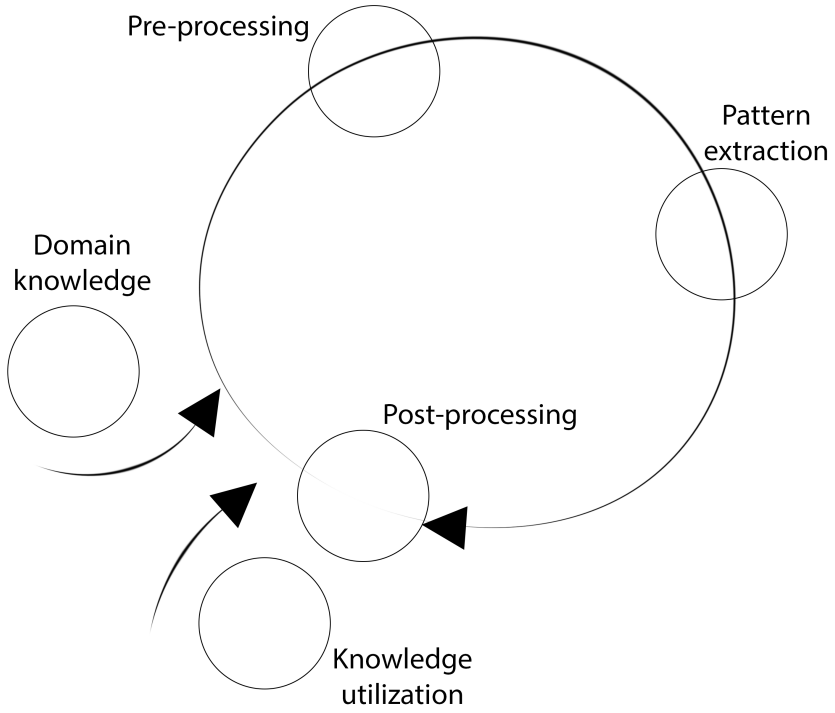


Figure 2 – Text mining organization cycle. Translated and adadpted from: (REZENDE, 2003).

- **Post-processing**: in the fourth step, the research must validate the quality and performance of the pattern extraction step. The researcher must use the correct sampling methods and performance metrics for machine learning. This step is further explained in the subsection 2.1.3;

- **Knowledge utilization**: in this step, the validated information extracted is used by the researcher and other stakeholders to utilize the knowledge obtained from the text. This knowledge can be used for modeling new text mining experiments within this organization cycle.

2.1.1 Pre-processing

In text mining, the pre-processing step aims to prepare and model the text data before the pattern extraction step. This process is usually modeled and executed before the pattern extraction as a whole since text data is sparse, increasing the computational costs from this step.

Early text mining works use modeling techniques like Bag of Words (BoW). More recently, neural language models that generate text embeddings like Word2Vec (MIKOLOV *et al.*, 2013) have been used for text mining modeling since they allow text to be represented within a fixed dimension representation space. Another way of generating text embeddings is by using contextual neural language models like BERT (DEVLIN *et al.*, 2018) since its structure allows for dynamic context and multilingual models. This dissertation focuses primarily on the use of information networks to add further contextual data to text data, which is discussed in

Section 2.2.

BoW

The BoW representation creates a $n \times d$ matrix, where n is the size of the lexicon, and d is how many documents make the dataset. Each document is then represented by a sparse vector with the appearance count of each token in that document (AGGARWAL; ZHAI, 2012; AGGARWAL, 2018; ZONG; XIA; ZHANG, 2021). This representation shows how sparse text data is since each vector will have the dimension of the lexicon, and most tokens will not be present in all documents.

Text mining operations, that may be used for other text mining scenarios, can be used to deal with the shortcomings of a traditional BoW (AGGARWAL; ZHAI, 2012):

- punctuation and stopwords removal: this technique aims to reduce the lexicon by removing unnecessary tokens from the vectors like punctuations, articles, and prepositions;
- stemming: is the technique of transforming all conjugations of the verbs to their radical, thus reducing dimensions by unifying words;
- lemmatization: is the technique of transforming all conjugations of the verbs to their infinitive form, thus reducing dimensions by unifying words;
- TF-IDF: instead of using the token frequency, the researcher must use the term frequency in the document divided by the number of documents that contain said token. This technique makes tokens that appear in many documents be penalized since they do not provide much discriminatory context when appearing in many documents.

Within a BoW representation, these operations help to reduce the dimension size but do not solve all the problems with the BoW representation. The vectors are still sparse, and different datasets will have different dimension sizes. Word embeddings allow the generation of fixed dimension dense vectors for each word (or document or sentence) (MIKOLOV *et al.*, 2013; GHAG; SHAH, 2015; DEVLIN *et al.*, 2018).

Word embeddings

Word embeddings are dense vectors generated through neural models for textual data representation. For example, Word2Vec generates dense vectors for each word in a phrase (MIKOLOV *et al.*, 2013). Figure 3 shows that these vectors allow each word to be positioned within the same vector space. This capability also allows for vector operation with words like: $(king - man) + woman \approx queen$.

Word2Vec and some other word embedding models are pre-trained in massive corpora. The structure Word2Vec pre-trains is called skip-gram. Figure 4 shows skip-gram's structure.

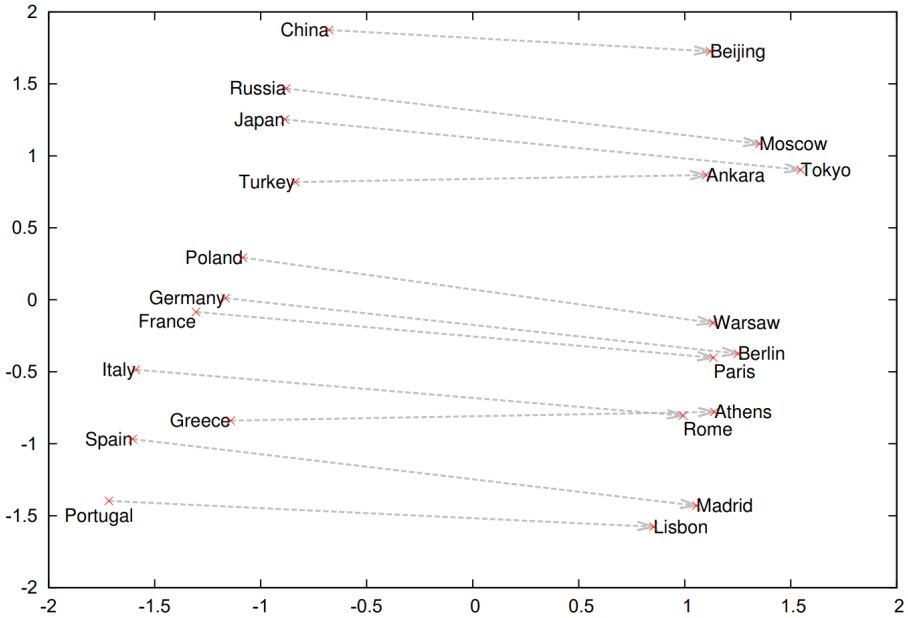


Figure 3 – PCA reduction of word embeddings from capitals and countries. Adapted from: (MIKOLOV *et al.*, 2013).

For example, skip-gram models learn by predicting the probability of a set of words $w_1, ..., w_T$ appearing when a specific word is an input. This training allows each word to receive a vector of "context" related to the corpora used in pre-training. The probabilities are obtained by applying a softmax activation function, as shown in Equation 2.1.

$$p(w_O|w_I) = \frac{\exp\left(v'_{w_O} \top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left(v'_{w_O} \top v_{w_I}\right)}, \tag{2.1}$$

where v_w is the input and v'_w the output embedding for the word w. However, this version of the skip-gram model is not scalable enough to be trained in billions of texts.

In order to allow scalability, the authors utilized a modified softmax equation called hierarchical softmax (MORIN; BENGIO, 2005). Hierarchical softmax is an approximation of the softmax through a binary tree. Each word w will have a path $L(w)$ representing its softmax value and $n(w, j)$ is the j-th on this path. Meanwhile $ch(n)$ represents a child node for any inner node n. The equation is represented at 2.2, reducing the final complexity to L.

$$p(w|w_I) = \pi_{j=1}^{L(w)-1} \mu\left([n(w, j+1) = ch(n(w, j))].v'_{n(w,j)} \top v_{w_I}\right) \tag{2.2}$$

The authors of Word2Vec also optimize the instance selection for training the model in a smaller portion of the lexicon. In this version the probability of $p(w|w_I)$ finding the target word w_O gets exchanged for a fake distribution $P_n(w)$ that represents k negative instances that the model must discriminate (Equation 2.3). So the subequation $P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$ defines how many words are discarded in order to sample rare and frequent words into training, where $f(w_i)$ is the word frequency and t the threshold.

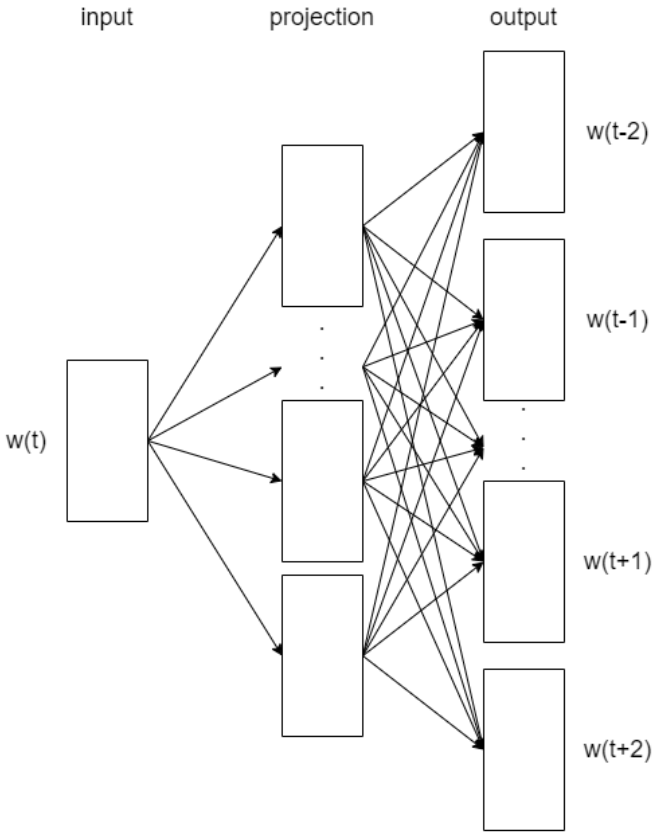


Figure 4 – Example of a skip-gram model. Adapted from: (MIKOLOV *et al.*, 2013).

$$\log \sigma(v'_{w_O} \top v_{w_I}) + \sum_{i=1}^{k} E_{w_i \sim P_n(w)}[\log \sigma(-v + w_i{}' \top v_{w_I})] \tag{2.3}$$

Although Word2Vec is pre-trained on most of the corpora, its embeddings are not capable of representing context dynamically. The lack of context means that words with multiple meanings might not be correctly represented in different scenarios. In order to overcome this limitation, other models for word embeddings have been developed.

The authors developed a model called Embeddings from Language Model (ELMo) (PETERS *et al.*, 2018). This method uses two Long Short-Term Memory (LSTM) neural networks to recognize the text sequentially. One LSTM "reads" the text from left to right and the other from right to left. This technique allows the words to be recognized and have their vectors positioned according to context.

ELMo's capability of generating context-aware embeddings allowed it to outperform other text embeddings in most text mining tasks. However, LSTMs are recurring neural networks, which makes ELMo less efficient in parallel execution. So Devlin *et al.* (2018) proposed a method called Bidirectional Encoder Representations from Transformers (BERT) that uses Transformers in its neural architecture. Transformers is a neural network architecture that uses attention mechanism and feedforward neural networks (Figure 5), developed by Vaswani *et al.*

(2017). They allow for better parallelization and faster training than recurrent neural networks, like LSTMs.

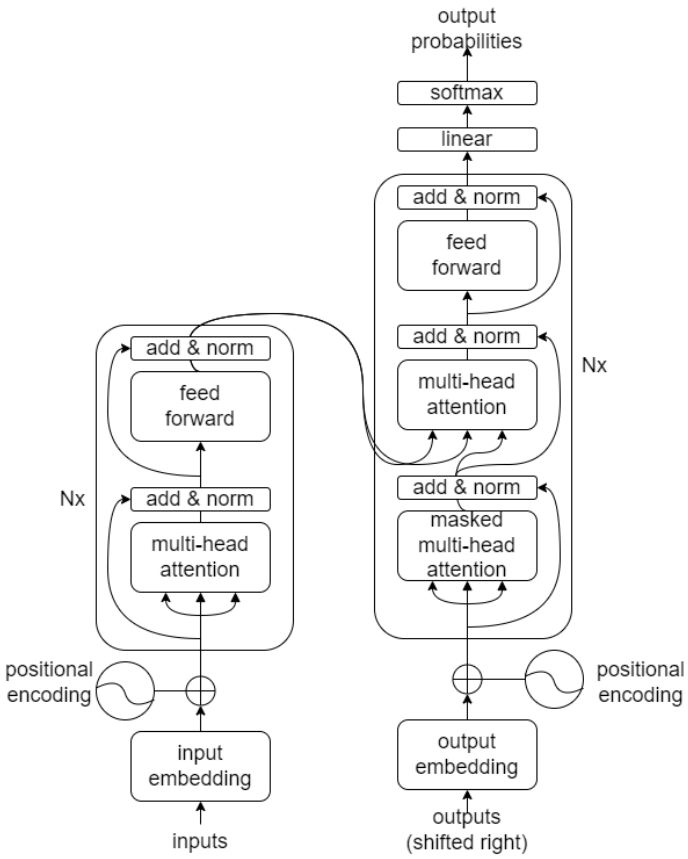


Figure 5 – Architecture encoder-decoder of a transformer. The encoder has $N = 6$ layers, where each sublayer has an attention head and a feedforward neural network for maintaining embedding positioning. The decoder also has $N = 6$ layers, but each sublayer has an extra attention head, re-positioned to avoid overfitting by learning word position. Adapted from: (VASWANI *et al.*, 2017).

BERT's transformers architecture allows for more extensive corpora on pre-training and a deeper contextual embedding than ELMO's. Transformers also allow another critical characteristic of BERT: fine-tuning. Figure 6 shows BERT's pre-training and fine-tuning architectures. There are also variations on BERT's basic architecture, allowing for more flexibility between performance and fast execution. These variations change the number of layers L, the hidden dimension H (which is also the resulting embedding vector dimension), and A, the amount of bi-direction attention heads:

1. $BERT_{BASE}$: $L = 12, H = 768, A = 12$;
2. $BERT_{LARGE}$: $L = 24, H = 1024, A = 16$;
3. $DistilBERT$: $L = 6, H = 512, A = 12$.

The architectures (1) and (2) are defined by Devlin *et al.* (2018) while architecture (3) is a reduced variation modeled, pre-trained, and evaluated to have nearly the same performance by Sanh *et al.* (2019).

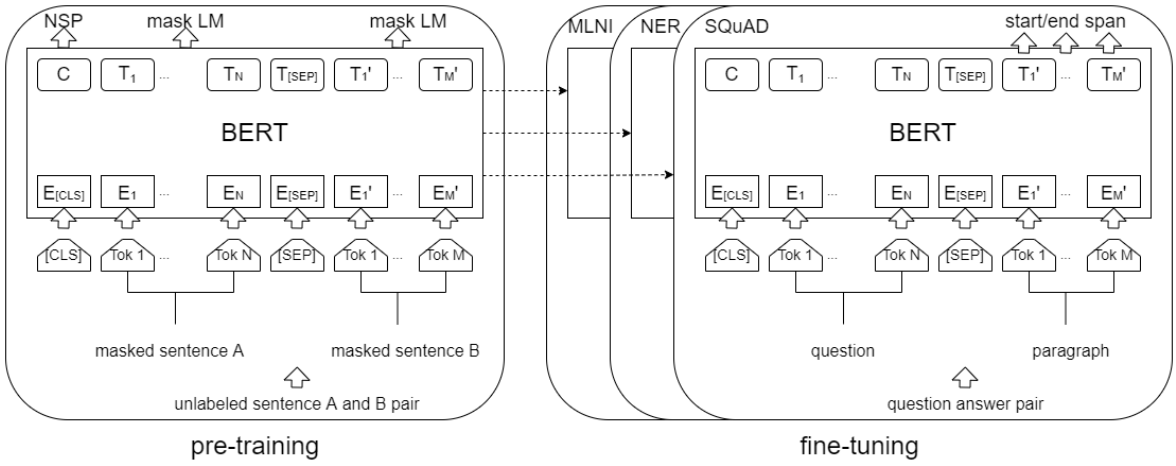


Figure 6 – Illustration of the pre-training and fine-tuning architectures of BERT. For pre-training, a bi-direction transformer is initialized and trained. This transformer is used to generate a general embedding for each input. Fine-tuning is initialized with the pre-training values and adds layers according to the task and necessities of the context training. Adapted from: (DEVLIN *et al.*, 2018).

BERT's pre-training can be performed with two different strategies. The first one is called, Masked-Language Modeling, where about 15% of the words from the corpora are hidden, and BERT must learn the word distribution to predict them correctly. The second one is called, Next Sentence Prediction, where selected phrases A are followed by others B and to every phrase A' BERT must find the following correct phrase B'.

Formally, for a token set $t_i = \{t_1, ..., t_k\}$ within each sequence $s = \{s_1, ..., s_n\}$ of a corpora n, the objective is finding the correct sequence s_i, as shown in Equation 2.4,

$$p(\bar{e}|\hat{e}) = \sum_{j=1}^{k} m_j p(t_j, c_j), \tag{2.4}$$

where $\hat{e}$ is a hidden token in the sequence s_i, and $\bar{e}$ are the masked tokens. The variable m_j is 1 when t_j is masked, otherwise it is 0. The variable c_j represents the context the token t_j belongs.

The bi-directional transformers within BERT must learn the distribution $p(t,c)$ of where the hidden token belongs in the sequence. So, the main equation is a conditional distribution of a token t in a context c, as shown in 3.1,

$$p(t|c) = \frac{exp(\mathbf{h}_c^\top \mathbf{w}_t)}{\sum_{t'} exp(\mathbf{h}_c^\top \mathbf{w}_{t'})}, \tag{2.5}$$

where $\mathbf{h}_c$ is an embedding vector for the context c and $\mathbf{w}_t$ an embedding vector for the token t. In $\sum_{t'} exp(\mathbf{h}_c^\top \mathbf{w}_{t'})$ the context embedding is normalized by all tokens t'. Finally, BERT outputs two embedding vectors $\mathbf{h}_c$ and $\mathbf{w}_t$.

After choosing a text representation, the text mining process can advance to the pattern extraction step.

2.1.2 Pattern extraction

Pattern extraction is critical for text mining. Within this step, the correct technique for a dataset can learn how to cluster, classify or predict important information from the ever-growing amount of text data. This dissertation focuses on machine learning (ML) for pattern extraction since it can be used to find patterns within vector spaces like embeddings. ML takes advantage of knowledge generalization but must be correctly implemented and maintained to avoid underfitting and overfitting. Underfitting is when the model cannot learn a pattern due to a lack of training data or model complexity. Overfitting is when the model learns the particularities of the training and is not able to generalize to other examples (AGGARWAL, 2018).

ML can be divided into three learning techniques for the models: (1) unsupervised, which usually clusters or identify patterns without additional data; (2) supervised, which makes use of a label for each example in the data to classify or predict new data; and (3) semi-supervised, which can learn patterns like supervised models but do not require the entire training data to be labeled (AGGARWAL, 2018).

More specifically, this subsection of the dissertation discusses artificial neural networks, which are used by methods and directly throughout the research. Neural networks are mostly used for classification, prediction or regression (LI *et al.*, 2018; YANG; ZHANG; HAN, 2019), but can be also used as encoders or decoders (AGGARWAL, 2018; MIKOLOV *et al.*, 2013; DEVLIN *et al.*, 2018), which enables their use for representation learning.

The perceptron (ROSENBLATT, 1958) is a neural network architecture for supervised learning. It learns to generalize in training by correcting errors compared to the labels (ALPAYDIN, 2020). Different neural networks can implement different weights, activation functions and error regularization functions (SATHYA; ABRAHAM, 2013). This subsection will elaborate on a few functions that can be used to implement a feed-forward neural network known as multilayer perceptron (MLP) and a recurring neural network known as long short-term memory (LSTM).

MLPs are formed by multiple layers of perceptrons, where each perceptron unit is connected to all the units on the next layer. Essentially an MLP is formed by three types of layers: (1) input layer; (2) hidden layers: layers in between that add complexity and the capabilities to generalize knowledge; and (3) output layer: layer where the desired output is converted to human-readable information (i.e., probability of a class, continuous number). Each unit contains a weight and a bias (Equation 2.6), and for the hidden layers and the output layer, an activation function that enables non-linear calculations (BOURLARD; KAMP, 1988):

$$unit_{c_j}(t) = \sum_u w_{c_j u} y^u, \tag{2.6}$$

where every j unity c has a weight w and a bias u for each received input y.

- **Sigmoid** (Equation 2.7): is a non-linear function that generates values between 0 and 1 (HAN; MORAGA, 1995). It is usually used as an activation function in hidden layers for smaller neural networks where all units must be active at all times. However, it can also be used in output layers for binary classifications with only one output unit:

$$F(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt; \tag{2.7}$$

- **ReLU** (Equation 5.3): is an activation function that returns the unit value when it is higher than 0 and 0 for everything else. This activation function is interesting when using deep neural networks, where it might be more interesting to let it learn when units are not necessary, meaning that it learns to turn units off (BROWNLEE, 2019):

$$F(x) = max(0,x); \tag{2.8}$$

- **SoftMax** (Equation 2.9): is an activation function usually used in an output layer and it outputs the probabilities for all units on the output layer (GOODFELLOW; BENGIO; COURVILLE, 2016). Softmax manages this behavior by being a derivative that considers the unit's index:

$$\frac{\partial}{\partial_{q_k}} \sigma(q,i) = \sigma(i,k)(\delta_{ik} - \sigma(q,k)) \tag{2.9}$$

LSTM networks use recurrent structures to create "memory" of patterns within sequential data inputs. Its memory control units allow the models to learn short and long sequences in their memory. To achieve this capability LSTM use two gate units (Equation 2.10) together with a common unit for every temporal step $t-1$. These gate units control when a unit can receive a signal $control_{in}$ and when to output it $control_{out}$ to the next layer or keep the value in the re-currency. In short, an LSTM unit is composed of three smaller units and is then called a memory cell (HOCHREITER; SCHMIDHUBER, 1997).

$$\begin{aligned} control_{in_j}(t) &= \textstyle\sum_u w_{in_j u} y^u(t-1) \\ controe_{out_j}(t) &= \textstyle\sum_u w_{out_j u} y^u(t-1) \end{aligned} \tag{2.10}$$

These two neural networks make use of a learning technique called back-propagation. Back-propagation is a supervised learning technique that updates the weights in the units by applying a minimizing function between the output and the label value for each example or batch of examples. Usually, the minimizing function is a type of gradient function. For an output t_i of an unit i and a real label o_i it updates the unit weight w_{ij}. For example a commonly used gradient for LSTMs is presented in Equation 5.2 from Williams and Peng (1990) where the time steps T, T' are also considered. However, this gradient function is based on Equation 2.12 and is a commonly used gradient for MLPs.

$$\Delta w^{E^{total}}(T',T) = \sum_{t=T^i+1}^{T} \Delta w^{E(t)} \rightarrow \Delta w_{ij} = -\alpha \frac{\partial E^{total}(T',T)}{\partial w_{ij}} \tag{2.11}$$

$$\delta_i = (t_i - o_i) \rightarrow \delta_i = \sum_{j=i+1}^{n_0} \delta_i w_{j,i} \rightarrow \Delta w_{i,j} = \delta_i o_j \tag{2.12}$$

As discussed in the subsection 2.1.1, LSTM networks are used in the ELMo model. The BERT model uses transformers, a neural network architecture that contains a feed-forward network and attention heads, which are simple recurrent networks.

2.1.3 Post-processing

The post-processing step is responsible for proposing and applying methods and techniques to evaluate the pattern extraction step correctly. Different domains, methods, and results need different techniques and metrics for a correct evaluation. For this dissertation, it is important to know and explain the sequential split, train-[validation]-test split, and the K-fold cross-validation techniques (AGGARWAL, 2018):

- **Sequential split**: this partition technique is used when a temporal sequence is needed for evaluating a model. For example, a time-series prediction task needs to be evaluated on predicting data from a point in time further than the data a model was trained on;
- **Train-[validation]-test split**: in this split for evaluating supervised models, the labeled data is divided into two or three splits: training, validation (optional), and test. All the data must be partitioned before training on the model. The training partition is where the model learns to generalize for a problem. The validation partition is only used if the problem is very susceptible to overfitting and labeled data is abundant since it allows for a check-up on new data performance at the end of every epoch. The test partition is where the final performance metrics are obtained. Usually, this split method is executed multiple times with different portions of the data on different splits to obtain performance metrics as an average of multiple executions;
- **K-fold cross-validation**: this technique divides the data in k equal partitions. Then the training test will be executed k times, with the training partition formed by $k-1$ smaller partition and one partition at a time being the test. This allows an average result of the performance metrics while testing with the entire dataset.

With the split technique selected, an appropriate performance metric must be chosen. This dissertation will discuss some classic performance metrics derived from the confusion matrix and ranking performance metrics used with recommendation systems:

- **Confusion matrix**: it is a structure that stores correct and wrong guesses within a set of rules: tp = true positive; tn = true negative; fp = false positive; and fn = false negative (POWERS, 2008);

	Positive prediction	Negative prediction
Positive label	tp	fn
Negative label	fp	tn

- **Accuracy** (Equation 2.13): it is simply the average between correct and wrong predictions (POWERS, 2008):

$$acc = \frac{tp + tn}{tp + tn + fp + fn} \tag{2.13}$$

- **Precision** (Equation 2.14): it is the ratio between positive correct predictions and all the positive predictions of the model (POWERS, 2008):

$$pre = \frac{tp}{tp + fp} \tag{2.14}$$

- **Recall** (Equation 2.15): it is known as the sensibility metric. It is the ratio between correct positive predictions and wrong negative predictions. So, recall measures examples that were positive and wrongly predicted negative (POWERS, 2008):

$$rcl = \frac{tp}{tp + fn} \tag{2.15}$$

- $F1$ (Equation 2.16): it is the harmonic average between precision and recall. This metric is upper limited by the accuracy, but it is less sensitive to unbalanced data (POWERS, 2008):

$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{2.16}$$

- MRR (Equation 2.17): mean reciprocal rank is a performance metric commonly used in recommendation systems where the user has a single correct document it desires. So the MRR metric averages the rankings for the correct document h for every recommendation $r \in R$. The final score for the MRR metric averages all the scores. The MRR metric can also be limited to score in the top k rankings and is known as $MRR@k$ (CRASWELL, 2009):

$$MRR = \frac{\sum \frac{1}{h}}{R} \tag{2.17}$$

- $Hits@k$ (Equation 2.18): is another performance metric used in recommendation systems where the user has a single correct document it desires. In the $Hits@k$ metric, every correct document within the top k rankings scores 1.00 and 0.00 for every other. The resulting score for the metric is a mean average of all the recommendations (DOCS, 2019):

$$Hits@k = mean\left(\begin{cases} 1.00 & h \leq k \\ 0.00 & h > k \end{cases}\right) \tag{2.18}$$

2.2 Information network representation learning

Networks are graphs with the capability of adapting nodes and edges to the complexity of the problem they are involved with (ROSSI, 2016; NEWMAN, 2010). For example, networks can be used with the following characteristics (NEWMAN, 2010):

- **Tech networks**: these networks are formed by physical connections between electronic equipments, like computer networks, telephone networks, and electrical grids;
- **Social networks**: these are networks where the nodes are profiles for people, businesses, songs, movies, and others. The edges are friendships, followers, likes, comments, and others. In particular, social networks tend to be sparse since most of the users do not interact with each other;
- **Biologic networks** are biologic networks involving biologic (or natural) components. For example, protein networks, metabolic reactions, and the food chain;
- **Information networks**: information networks or knowledge graphs are data networks. Data and its relations can be modeled by nodes and edges, which makes them adaptable for storing and indexing data. These networks are tightly integrated with the research in this master's dissertation.

Information networks can be formally represented as a triple $N =< O, R, W >$ where: O is the set of objects; R is the set of relations, or connections, between objects; and W is the set of weights for each relation. There are different types of information networks. These different characteristics might coexist, for example, a network can be undirected, weighted and homogeneous at the same time defined by different rules for the set of triples, like (ROSSI, 2016):

- **Undirected networks**: in these networks whenever there is a relation r_{o_i,o_j}, a relation r_{o_j,o_i} automatically exists, meaning that every connection comes and goes;
- **Directed networks**: unlike undirected networks, whenever a relation r_{o_i,o_j}, a relation r_{o_j,o_i} does not automatically exists, meaning that a specific flow must be followed when navigating and structuring the network;
- **Unweighted networks**: in these networks, all the relations r_{o_i,o_j} have the same weight w_{o_i,o_j}, meaning that they do not have this differentiating factor between them;
- **Weighted networks**: unlike unweighted networks these networks have different weights w_{o_i,o_j} for different relations. This metadata can differentiate types of objects in their relations or even by relation, depending on the complexity.

- **Homogeneous networks**: in these networks, all objects $o \in O$ have only one type of data. This structure is easier to treat in machine learning scenarios, but it is also more limited in regards to the complexity of the data it can model;
- **Heterogeneous networks**: unlike homogeneous networks in these networks, objects $o \in O$ have more than one type of data. These networks are more flexible when modeling real-world data but more complex to generalize in machine learning models. This type of information network is the method's focus in this master's dissertation.

Heterogeneous networks also have different known configurations that can not coexist and might allow for standards ways of processing, like:

- **Bipartite networks**: these heterogeneous networks are composed by two types of objects $o \in O$ and its relations r_{o_i,o_j} are only possible when o_i, o_j have different types of data;
- **k-partite networks**: these heterogeneous networks are like an extension of bipartite where the objects $o \in O$ have k different types of data and the relations r_{o_i,o_j} are only possible when o_i, o_j have different types of data;
- **Star networks**: in these heterogeneous networks, a particular typed object $o \in O$ is the only data type that can connect to the rest and between each other. For example, a network of papers where each can be related to others, but only a paper can have characteristics like authors, published date, publisher, and others.

Although heterogeneous networks allow for modeling text data, they also need processing before applying to text mining ML methods and techniques. The following subsections will explain methods and techniques essential for this master's dissertation: regularization methods and network embedding.

2.2.1 Regularization methods

Usually, regularization methods for information networks are designed as semi-supervised classification methods. These methods use the regularization technique to transfer labels from neighboring objects in search of a local and/or global equilibrium (ENGELEN; HOOS, 2020). These methods consider that objects with the same label will be connected somehow. This subsection will discuss some regularization methods for homogeneous and heterogeneous networks.

Semi-supervised Learning Using Gaussian Fields and Harmonic Functions

The Semi-supervised Learning Using Gaussian Fields and Harmonic Functions (GFHF) method uses harmonic functions to minimize a regularization function in homogeneous information networks. This was designed for semi-supervised classification and was evaluated and validated in gaussian fields (ZHU; GHAHRAMANI; LAFFERTY, 2003).

GFHF's aims to correctly classify nodes with unknown labels u ($u = x_1, x_2, ..., x_u$) from labeled nodes l ($l = (x_1, y_1), (x_2, y_2), ..., (x_l, y_l)$), where the total number of nodes n, $n = l + u$, and usually $l < u$ in a binary classification problem where each label $y \in \{0, 1\}$. GFHF uses a symmetric matrix W that stores all weights from the relations of a homogeneous information network after going through the Equation 2.19, where m is the number of characteristics and σ the gaussian opening.

$$w_{ij} = exp\left(-\sum_{d=1}^{m} \frac{(x_{id} - x_{jd})^2}{\sigma_d^2}\right) \tag{2.19}$$

A quadratic energy function $f : V \to \mathbb{R}$ generates a continuous number for regularizing throughout the network. The function f is described in Equation 2.20:

$$E(f) = \frac{1}{2}\sum_{i,j} w_{ij}(f(i) - f(j))^2 \tag{2.20}$$

With the continuous number outputted by function f is used to generate the probabilities of a label for each node $p_\beta(f) = \frac{e^{-\beta E(f)}}{z_\beta}$, where β is the inverse of a temperature value and Z_β is the partition function $Z_\beta = \int_{f|L=f_l} exp(-\beta E(f))df$ to normalize previous labeled nodes f_l values. In short, GFHF (Equation 2.21) is composed by the minimization of the harmonic functions:

$$f(j) = \frac{1}{d_j}\sum_{i \sim j} w_{ij} f(i), para\, j = l+1, ..., l+u \tag{2.21}$$

Learning with Local and Global Consistency

The Learning with Local and Global Consistency (LLGC) method was inspired by GFHF. The authors Zhou *et al.* (2004) propose a method that considers a set of nodes $X = \{x_1, ..., x_l, ..., x_u, ..., x_n\}$ where the labeled nodes $y \in Y$, $Y = \{y_1, ..., y_l\}$ and $u \in U$, $U = \{\}$ and usually $y \leq u \leq n$. LLGC is an iterative method structured as follows:

1. a weight matrix W is generated with the edges relations with the equation $W_{ij} = exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$ if $i \neq j$ and $W_{ii} = 0$;

2. using the matrix W, $S = D^{-1/2}WD^{-1/2}$ is extracted where D is a diagonal matrix and the element (i, i) is equal to the i-eth line;

3. the results are then iterated over $F(t+1) = \alpha SF(t) + (1-\alpha)Y$ and α is a parameter between $(0, 1)$ that determines the amount of regularization from the neighboring nodes;

4. after all that F^* is the ceilling from the sequence $\{F(t)\}$, allowing the distribution of a probability of a label $y_i = argmax_{j \leq c} F^*_{ij}$ for each node x_i.

The iterative portion from LLGC can be seen activating values throughout the network. This allows the method to search for a global equilibrium between labeled nodes and the relations from the network. The final regularization function F, which is minimized by this process, can be seen in Equation 2.22:

$$Q(F) = \frac{1}{2}\left(\sum_{i,j=1}^{n} W_{ij}||\frac{1}{\sqrt{D_{ii}}}F_i - \frac{1}{\sqrt{D_{jj}}}F_j||^2 + \mu\sum_{i=1}^{n}||F_i - Y_i||^2\right), \tag{2.22}$$

where the first portion calculates the distances from connected nodes and their regularized labels, while the second portion normalizes the results with the existing labels values.

Graph Regularized Transductive Classification on Heterogeneous Information Networks

The authors Ji *et al.* (2010) proposed a regularization method for HINs called Graph Regularized Transductive Classification on Heterogeneous Information Networks (GNetMine). HINs are more challenging to generalize in ML methods, namely: (1) the complexity of having different types of data and using this metadata; (2) the difficulty of prioritizing these different types in the generalization process; and (3) the difficulty to find labeled data to use in supervised models. So GNetMine focuses on solving (1) and (2), while (3) is covered by it being a semi-supervised model.

An HIN $N =< O, R, W >$ has $t > 1$ different types of nodes. GNetMine uses sub-networks that contain relations of different types $N' \subseteq N$ for its equations, where $N' =< O', R', W' >$, $O' \subseteq O$, $R' \subseteq R$ and $W' \subseteq W$. It regularizes to classify unknown labels through a connections matrix C_{ij} of size $n_i \times n_j$ that corresponds to a graph of relations G_{ij} where o_i and o_j, $i, j \in \{1, ..., t\}$. Since i and j can have the same size, the weights in this matrix are defined as $C_{ij,pq}$, representing the relations $< x_{ip}, x_{jq} >$. This matrix receives the weights W when x_{ip} and x_{jq} are connected. The labels from k dimensions are codified in a vector $y_i^{(k)} = [y_{i1}^{(k)}, ..., y_{in_i}^{(k)}]^T \in \mathbb{R}^{ni}$ for each x_{ip} previously labeled.

Through the different connection matrix C_{ij} is defined a diagonal matrix D_{ij} of size $n_i \times n_i$, where the element type (p, p) is the sum of the the p-eth line of C_{ij}. This ensures the function $f_i^{(k)}$ has consistency with the types of the relations and the existing labels. For this regularization method the relations matrix and their diagonals are separated, where $D_{ij,pp}$ and $D_{ji,qq}$ are respectively the (p, p)-eth of D_{ij} and the (q, q)-eth of D_{ji}. The relation matrix and the diagonals are simplified as a regularization expression $S_{ij} = D_{ij}^{(\frac{-1}{2})} R_{ij} D_{ji}^{(\frac{-1}{2})}, i, j \in \{1, ..., t\}$. This regularization expression allows the normalization of labels considering the different types of relations to apply the Equation 2.23.

$$\begin{aligned} J(f_1^{(k)}, ..., f_m^{(k)}) &= \sum_{i,j=1}^{m} \lambda_{ij}((f_i^{(k)})^T f_i^{(k)} + (f_j^{(k)})^T - 2(f_i^{(k)})^T S_{ij} f_j^{(k)}) \\ &+ \sum_{i=1}^{m} \alpha_i (f_i^{(k)} - y_i^{(k)})^T (f_i^{(k)} - y_i^{(k)}). \end{aligned} \tag{2.23}$$

Allied to the regularization the parameters λ_{ij} and α_i control respectively, the amount of value to the relation between different types and the confidence on existing values, where $0 \leq \{\lambda_{ij}, \alpha_i\} < 1$. GNetMine can also be applied as an iterative method, described as follows:

1. for $\forall k \in \{1, ...K\}, \forall i \in \{1, ..., t\}$ the confidence parameters $f_i^{(k)}(0) = y_i^{(k)}$ and $iter = 0$ are initialized;
2. for each $f_i^{(k)}(iter)$ is calculated: $f_i^{(k)}(iter+1) = \frac{\sum_{j=1, j\neq 1}^{m} \lambda_{ij} S_{ij} f_i^{(k)}(iter) + \alpha_i y_i^{(k)}}{\sum_{j=1, j\neq 1}^{m} \lambda_{ij} + 2\lambda_{ii} + \alpha_i}$;
3. repeat step 2 for each $iter = iter + 1$ until $f_i^{(k)*} = f_i^{(k)}(iter)$ achieves a value below a threshold for each i;
4. for the p-eth label associated to the type X_i for each $i \in \{1, ..., t\}$, through $c_{ip} = argmax_{1 \leq k \leq K} f_{ip}^{(k)*}$, where $f_i^{(k)*} = [f_{i1}^{(k)*}, ..., f_{in_i}^{(k)*}]$.

Label Propagation on Heterogeneous Information Networks

Based on LLGC (ZHOU *et al.*, 2004) the authors Rossi, Lopes and Rezende (2014) modeled a regularization method for semi-supervised classification in text HINs called Label Propagation on Heterogeneous Information Networks (LPHIN). More specifically LPHIN was first modeled to be executed in bipartite HINs where the two types of nodes correspond to documents $D = \{d_1, ..., d_m\}$ and tokens $T = \{t_1, ..., t_m\}$ to represent relations $D \rightarrow T$.

LPHIN is an iterative method that converges through the equation $F = PF$, where F is a matrix of label information and P is a matrix of relation probabilities between tokens and documents. The probability matrix P is calculated in two different ways: the probability of a document d_i to be connected to a token t_j (Equation 2.24); and the probability of a token t_j to be connected to a document d_i (Equation 2.25).

$$p_{d_i,t_j} - \frac{w_{d_i,t_j}}{\sum_{d_k \in D, w_{d_k,t_j} \in W} w_{d_k,t_i}} \tag{2.24}$$

$$p_{t_j,d_i} - \frac{w_{d_i,t_j}}{\sum_{t_k \in T, w_{d_i,t_k} \in W} w_{d_i,t_k}} \tag{2.25}$$

Then the matrixes are divided into labeled documents (D^L), unlabeled documents (D^U), and tokens (T). Then the equivalent matrix of labels F and connection probability P are used as multipliers to regularize the network (Equation 2.26):

$$\begin{bmatrix} F_{D^L} \\ F_{D^U} \\ F_T \end{bmatrix} = \begin{bmatrix} P_{D^L D^L} & P_{D^L D^U} & P_{D^L T} \\ P_{D^U D^L} & P_{D^L D^L} & P_{D^U T} \\ P_{T D^L} & P_{T D^U} & P_{TT} \end{bmatrix} \begin{bmatrix} F_{D^L} \\ F_{D^U} \\ F_T \end{bmatrix} \tag{2.26}$$

The regularization process propagates the labels (F_{D^L}) to unlabeled objects (F_T, F_{D^U}) iteratively through the equations 2.27 and 2.28:

$$F_T^{(n)} = \sum_{i=0}^{n-1} (P_{D^U T} P_{T D^U})^i P_{T D^L} Y^L + (P_{T D^U} P_{D^U T})^{n-1} P_{T D^U} F_{D^U}^{(0)} \tag{2.27}$$

$$F_{D^U}^{(n)} = \sum_{i=0}^{n-1} (P_{D^U T} P_{T D^U})^i P_{D^U T} P_{T D^U} Y^L + (P_{D^U T} P_{T D^U})^n F_{D^U}^{(0)} \tag{2.28}$$

The matrix is normalized line by line, and the sum operations return a γ between 0 and 1. As LPHIN converges, γ tends to 0. At the end of the process a final label c_l for each document d_i is outputted as the highest probable $Pr[c_l]$ by Equation 2.29:

$$class(d_i) = \arg \max_{1 \leq l \leq |c|} Pr[c_l] . \frac{f_{i,l}(D^U)}{\sum_{d_j \in D} f_{j,l}(D)} \tag{2.29}$$

2.2.2 Network embedding methods

Network embedding methods aim to translate topology, types, and other data contained within information networks to dense vectors. They also aim to overcome the following challenges (CUI *et al.*, 2018):

- **Computational cost**: networks stores local and global relations within the data, but the computational cost of processing the structure for pattern extraction is very high therefore, it is not scalable; and
- **Difficult to parallelize**: in order to extract patterns from information networks, sequential access is usually necessary, which limits the steps in which parallel execution is possible.

This subsection presents some network embedding methods for information networks: DeepWalk, Node2Vec, Struc2Vec, Metapath2Vec, Large-scale Information Network Embedding (LINE), and Graph Convolutional Network (GCN).

DeepWalk, Node2Vec, Struc2Vec and Metapath2Vec

This subsection explains the basics of the network embeddings methods derived from Word2Vec. The methods DeepWalk, Node2Vec, Struc2Vec and Metapath2Vec, are used as baselines in this dissertation. All these methods use random or biased walks through the network to sample "phrases" and train a skip-gram model. It will be mentioned and explained whenever a method changes these fundamentals to obtain more information and add complexity.

DeepWalk (PEROZZI; AL-RFOU; SKIENA, 2014) is the classical adaptation of Word2Vec to network embedding and uses random walks and a skip-gram model. It aims to embed the

network's topology in dense $\mathbb{R}$ vectors. It accepts walk length and final dimensions of the vectors as hyperparameters.

The authors Grover and Leskovec (2016) adapted the classic DeepWalk to accept two extra hyperparameters, p and q, and created a method called Node2Vec. These hyperparameters modify the random walk in each step. The hyperparameter p modifies the probability of revisiting a node. Furthermore, the hyperparameter q modifies the probability of choosing a neighbor or a neighbor from a neighbor.

Metapath2Vec (DONG; CHAWLA; SWAMI, 2017) adapts the classic DeepWalk to use the edge type metadata in HINs. For that, the authors transformed the classic random walk into a metapath-based random walk. A metapath is formally defined as a cyclic sequence of edge types that must be followed when exploring a HIN. For example, when using the metapath EAE to explore a HIN, where E extends for event nodes and A for actor nodes, a metapath-based random walk would have to choose a random E node, and from there, a random A node and so on. These metapaths allow domain knowledge about the dataset relation types to be incorporated into the network embedding generation process. There are also automatic metapath generation algorithms when the dataset is too complex or there is insufficient domain knowledge for defining one at the beginning of the process.

The Struc2Vec (RIBEIRO; SAVERESE; FIGUEIREDO, 2017) adapts the classic DeepWalk to embed communities of nodes within the networks. Unlike Node2Vec and Metapath2Vec, which modify existing structures from DeepWalk, Struc2Vec adds another step to the pipeline. Before applying the random walk, it constructs a multi-layered graph, where each layer represents a level on the graph concerning a random node. This allows Struc2Vec to recognize communities of nodes connected to a single one within the same level.

One of the most defining techniques for these methods is random walks. A random walk W_{v_i} starts on the node v_i and proceeds in the following edges creating a path $W^1_{v_i}, \ldots, W^k_{v_i}$ that contains context from neighboring nodes as shown in Equation 2.30.

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{v,r}}{Z} & \text{if}(v,x) \in R \\ 0 & \text{if}(v,x) \notin R \end{cases} \tag{2.30}$$

Random walks are sensitive to the local context in information networks. They also allow for parallel execution since the walkers can be executed independently. This behavior allows for information to be retrieved from massive information networks. DeepWalk and Struc2Vec use the classic random walk technique, while Node2Vec and Metapath2Vec add some constraints to try and retrieve more data from the network.

Node2Vec uses the concepts of depth and breadth search to insert a bias on the network. When the random walk tends to breadth, search nodes on the same level will be explored first, which helps to extract local neighborhoods. However, when the random walk tends to

depth, search nodes on other levels will be explored first, helping the method better map global neighborhoods. So this biased walk can be defined as $\alpha_{pq}(t,x)$, where (t,x) is the probability of transitioning on the edges (v,x) (Equation 2.31).

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if} d_{tx} = 0 \\ 1 & \text{if} d_{tx} = 1 \\ \frac{1}{q} & \text{if} d_{tx} = 2 \end{cases} \tag{2.31}$$

The parameter p and q control the tendencies from the bias. The smaller the p higher is the probability of re-visiting nodes. The parameter q controls the probability of choosing breadth or depth search, where $q > 1$ tends to breadth search and $q < 1$ tends to depth search.

Metapath2Vec uses metapaths to guide random walks. In a HIN $N = (O,R,W)$ a set of metapaths $P : o_1 \rightarrow^{R_1} o_2 \rightarrow^{R_2} ...o_s \rightarrow^{R^s} V_{s+1}$ has a probability of transitioning to other nodes p defined in Equation 2.32.

$$p(v^{i+1}|v_s^i,P) = \begin{cases} \frac{1}{|N_{s+1}(v_s^i)|} & (v^{i+1},v_s^i) \in R, \phi(v^{i+1}) = s+1 \\ 0 & (v^{i+1},v_s^i) \in R, \phi(v^{i+1}) \neq s+1 \\ 0 & (v^{i+1},v_s^i) \notin R \end{cases} \tag{2.32}$$

This equations guarantees that the walk will follow a cyclic metapath $v^{i+1} \in V_{s+1}$. Although metapaths allow Metapath2Vec to bias its embedding vectors towards certain typed relations, this technique requires extra processing to achieve better performance. At the same time, it may hurt performance compared to a traditional random walk if the metapaths are not well chosen.

LINE

The authors Tang *et al.* (2015) developed a network embedding method capable of mapping in a function $f_G : V \rightarrow R^d$, local $u \rightarrow v$ relations with weight w_{uv} and global $u \rightarrow ... \rightarrow v$ with weight $(w_{u1},...,w_{uv})$ for each node $v \in V$. The method is called LINE, and it aims to tackle this problem in networks with millions of nodes and billions of edges. The method can also be applied in either directional or undirectional networks and weighted and unweighted networks.

So the LINE method captures probabilities for local and global relations are optimized separately. The local relations are capture by the equation O_1 presented in 2.33.

$$O_1 = -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i,v_j) \tag{2.33}$$

For global relations, LINE considers the node and its generated context formed by the connections of the nodes. For that LINE assumes a distance $d(.,.)$ between the distribution $p_2(.|v_i)$ that captures this context and the distribution $\hat{p_2}(.|v_i)$ that is the current global embedding

vector in order to achieve an equilibrium. The Equation 2.34 is used as the weights for edges, and the global probability is minimized.

$$O_2 = -\sum_{(i,j)\in E} w_{ij} \log p_2(v_i|v_j) \tag{2.34}$$

Both functions are optimized separately considered the objective function presented in Equation 2.35. In order to make the training scalable, LINE uses negative sampling to find the correct edge (i, j) in a noisy distribution, where K is the number of negative edges.

$$\log \sigma(\vec{u}_j^{|T} . \vec{u}_i) + \sum_{i=1}^{K} E_{v_n} P_n(v)[\log \sigma(-\vec{u}_j^{|T} . \vec{u}_i)] \tag{2.35}$$

After that, LINE is minimized through a mini-batch. The resulting embeddings are formed by concatenating the local and global embeddings.

GCN

GCN is a semi-supervised classification method that uses existing label information and local relations (KIPF; WELLING, 2017) for generating a function $f(X,A)$ trained in labeled nodes L_0. First, GCN propagates the nodes on a convolutional network and applies spectral filters (HAMMOND; VANDERGHEYNST; GRIBONVAL, 2011) to learn information from the network. The spectral convolutions are a signal multiplication $x \in \mathbb{R}^N$ with a filter $g_\theta = diag(\theta)$, where $diag$ is the diagonal of a Laplacian matrix $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T$. The Laplacian matrix used in GCN is an approximation calculated through the Chebyshev polynomials 2.36 $T_k(x)$ until the K-eth order to reduce computational cost.

$$g_\theta^{'} * x \approx \sum_{k=0}^{K} \theta_k^{'} T_k(\tilde{L}) x \tag{2.36}$$

For the convolutions, GCN utilizes a neural network of multiple layers, using the spectral filters (DEFFERRARD; BRESSON; VANDERGHEYNST, 2016). The convolutions allow a deeper capability of modeling a non-linear function to classify the nodes. For the classification output, GCN uses a softmax layer. A fixed dimension embedding vector can be obtained from the weights of each unit in the penultimate layer.

2.3 Applications

This section presents related works of methods and techniques related to embedding propagation. The subsections contain works related to the basis of the method: embedding propagation, network embedding, and regularization. They also contain works related to the

applications used to evaluate and validate the proposed methods: event analysis and graph completion.

Embedding propagation methods

The authors Duran and Niepert (2017) proposed a semi-supervised embedding propagation method for information networks. Their method uses the BoW representation to start up the nodes' embeddings. Nodes with the same label are connected and refined in pairs, and a global equilibrium between the labels and nodes' distances is the target by applying a gradient function. In the end, each node receives a unique embedding. Even though this method allows for the same vector space to be maintained, the BoW representation can only encode one language at a time and has its challenges. Another disadvantage is that by refining between pairs, the computational cost is quadratic. The method outperformed random walk baselines in some scenarios only by around 1% $F1$.

The authors Yang, Zhang and Han (2019) proposed a semi-supervised neural embedding propagation method. The method generates embeddings through a two-stage propagation: it initializes the vectors through an MLP that uses node positioning; and it propagates and adjusts the values further through a series of MLPs that consider nodes and distances in different random walks. Even though it is an embedding propagation method, it does not consider other features to initialize the embeddings, which does not allow for pre-trained embedding's general knowledge to be used. Multiple MLPs have a high computational cost for training and execution compared to a graph neural network model like GCN. The method outperformed all baselines, in all datasets but by little margin of less than 1% in most scenarios.

Network embedding methods

The authors Yu *et al.* (2020) propose a semi-supervised network embedding method based on using paths for different types in a heterogeneous information network for training a variational auto-encoder. The method has shown better performance than other methods in the literature, indicating that variational auto-encoders can generalize knowledge for heterogeneous information networks. However, it is not optimized for large-scale or sparse networks, which makes it difficult to be used in the real world. The proposed method Rich Heterogeneous Information Preserving Network Representation Learning (HIRL), consistently outperforms baselines by around 5% AUC in all scenarios.

The authors Gui *et al.* (2017) propose a network embedding method called HEBE designed primarily for event networks. This method uses the concept of hyperedges to aggregate edges of the same type, effectively creating communities of events between them. The model also optimizes its execution using sub-sampling and negative-sampling and can be used for hyperedge prediction. The HEBE model in its HEBE-PE and HEBE-PO variants outperforms

baselines in two databases by at least 2% *AUC* and *accuracy* to the second best performer.

The authors Setty and Hose (2018) modified Node2Vec's biased walk to use Jaccard's similarity (JACCARD, 1901) in place of the probabilities to change the next step between DFS and BFS. The method called Event2Vec takes advantage of this biased walk to better explore different types of event networks without manually assigned meta-paths. Although Event2Vec can use the same devices for scalability as Node2Vec obtaining the Jaccard similarity for every step adds a high computational cost. Event2Vec achieved better performance than Node2Vec in both $Precision@k$ and $Recall@k$ metrics by at least 0.4 points.

Regularization methods

The authors Rossi, Lopes and Rezende (2017) proposed a semi-supervised classification method based on a graph construction method called GBILI (BERTON; LOPES, 2014) and a label propagation method called LLGC (ZHOU *et al.*, 2004). The proposed method constructs a new sparse network where each labeled node is connected to its neighbors. The new sparse network propagates labels with the regularization function to achieve local and global equilibrium. The method has achieved good performance compared to other classification methods. It has increased $F1$ performance within 10% in some scenarios.

The authors Santos *et al.* (2020) proposed a regularization method called Heterogeneous Event Network Regularization in Two-stages (HENR^2). As the name of the method implies, it applies the regularization in two stages. In the first stage, the relations are regularized and receive an entropy value corresponding to their value in the network. In the second stage, the regularization uses the fixed weight of the relations. The two-stage method helps reduce noise within the regularization but can also propagate errors from previous iterations forward. HENR^2 performed better than baselines with a 2% increase of $F1$ in some scenarios.

Event analysis

The authors Ning *et al.* (2016) proposed a method for predicting future occurrences of protest events using news text data. They use super-bags, where each bag is a news text embedding and a collection of news from a particular period in time is a super-bag. The super-bags are then fed to three different models: (1) it calculates the similarity between super-bags in order to determine precursor events; (2) it calculates the probability of a new protest occurrence from the precursor events; and (3) is a multi-class prediction of how many participants will attend said protest. The model shows great performance in the experiments executed. It outperformed baselines with nearly 10% more accuracy and $F1$ in some cases. However, they were limited to specific datasets of certain regions, and the models can not accept explicit features from events like locations and organizations involved.

The authors Deng, Rangwala and Ning (2019) model the protest events in a dynamic

graph of import tokens from each event. These graphs also allow for access to the embeddings of each token. The graph is then applied to a GCN model that aims to predict a temporal relation t between the protest events. The method has achieved good performance, outperforming baselines $F1$ results in every scenario by 2% in some cases. However, the experiments were executed in specific datasets of protest events in certain regions.

The authors Deng, Rangwala and Ning (2020) proposed a multi-task model for event and actors prediction by modeling the data in dynamic graphs and using graph completion techniques. The modeled graphs are dynamic because they allow the insertion of new data into the pipeline of GCN to predict links to related events and actors in the graph. Although this method allows for dynamic insertion, the GCN model necessitates a percentage of labeled data to achieve good performance. The model has only experimented with particular datasets of protests in certain regions. However, the proposed model achieved excellent performance compared to baselines with the least improvement in $F1$ reaching 3.1%. The model also increased $Hits@k$ performance with as much as 27.5% in one scenario.

Graph completion and link prediction

The authors Ozcan and Oguducu (2019) proposed a link prediction method for heterogeneous information social networks called Multivariate Time Series Link Prediction. Besides considering different types of relations, the method also considers local and global distances and temporal data. The method creates vectors that are concatenated according to the metadata. The links are predicted by receiving these concatenated vectors and measuring similarity. The method achieved good performance when predicting global relationships but underperformed baselines for local relationships.

The authors Chen *et al.* (2018) propose a network embedding method for HINs based on project metric embedding. This method generates different embedding vectors for different network relations types. These vectors are then used to obtain local and global distance measures between nodes to aggregate the different embedding vectors. The method performs better than other literature network embedding methods, especially on link prediction tasks. It achieved at least 7% more AUC than other baseline methods within all binary link prediction scenarios.

The authors Zhou *et al.* (2018) proposed a method for link prediction in a social network by combining network embedding methods to a probabilistic model from offline movement in the social network. The method called Vec2Link combined user check-in history with the other network data to achieve better performance. It achieved at least 5% more AUC than baseline network embedding methods, combined with different similarity measures for link prediction.

2.4 Concluding remarks

This chapter presents fundamental concepts on text mining and HIN representation learning. Related applications on embedding propagation, network embedding, regularization, event analysis, and graph completion were also presented. These applications are essential for this master's dissertation because they relate directly to the proposed method or the domains used in the evaluation experiments.

The proposed method in this master's dissertation uses regularization for embedding propagation on HINs modeled after text data. The method outputs unique embeddings for all nodes, considering the initial text embeddings' vector space, allowing the dynamic insertion of new nodes. The proposed method is presented in the following chapter.

CHAPTER

3

EMBEDDING PROPAGATION OVER HETEROGENEOUS INFORMATION NETWORKS

3.1 Motivation

As introduced in Chapter 1, accessible text data is exponentially growing because of the internet. Because humans are able to consume information from unstructured data sources, machines require text data to be cleaned and transformed for machine readability. Usually texts, sentences and/or words are transformed into vectors before applying extraction methods.

HINs can also be used to structure text and even enrich data by explicitly connecting characteristics or combining the data with other HINs. In Chapter 2 presents network embedding methods for extracting vectors from these HINs. However, regular network embedding methods are not able to benefit from existing neural language models knowledge when applied to HINs with textual data in some nodes.

This master's dissertation presents an embedding propagation method of text embeddings in HINs with textual data in some objects to generate new embedding vectors that contain related data and metadata. The proposed embedding propagation method uses a regularization function to propagate BERT embedding vectors instead of labels. This allows the embedding vectors to contain data directly from the text and other relations. The proposed method also maintains BERT's space vector, allowing dynamic generation of newly added nodes. More details of the proposed method are discussed below.

3.2 Method: Embedding propagation over heterogeneous networks (EPHEN)

Considering a heterogeneous information derived from text data represented as $N = (O, R, W)$, where O is a set of typed objects $O = \{O_r \cup O_c\}$ defined as: O_r raw text (documents, sentences, others); and O_c complements (dates, objects involved, others). The objective is to define an embedding propagation function $f : O \rightarrow \mathbb{R}^d$, where for each node $o \in O$ an embedding vector $\mathbf{f_o} \in \mathbb{R}^d$ is generated. The steps from the proposed method illustrated in Figure 7 are described in the subsections 3.2.1, 3.2.2 and 3.2.3.

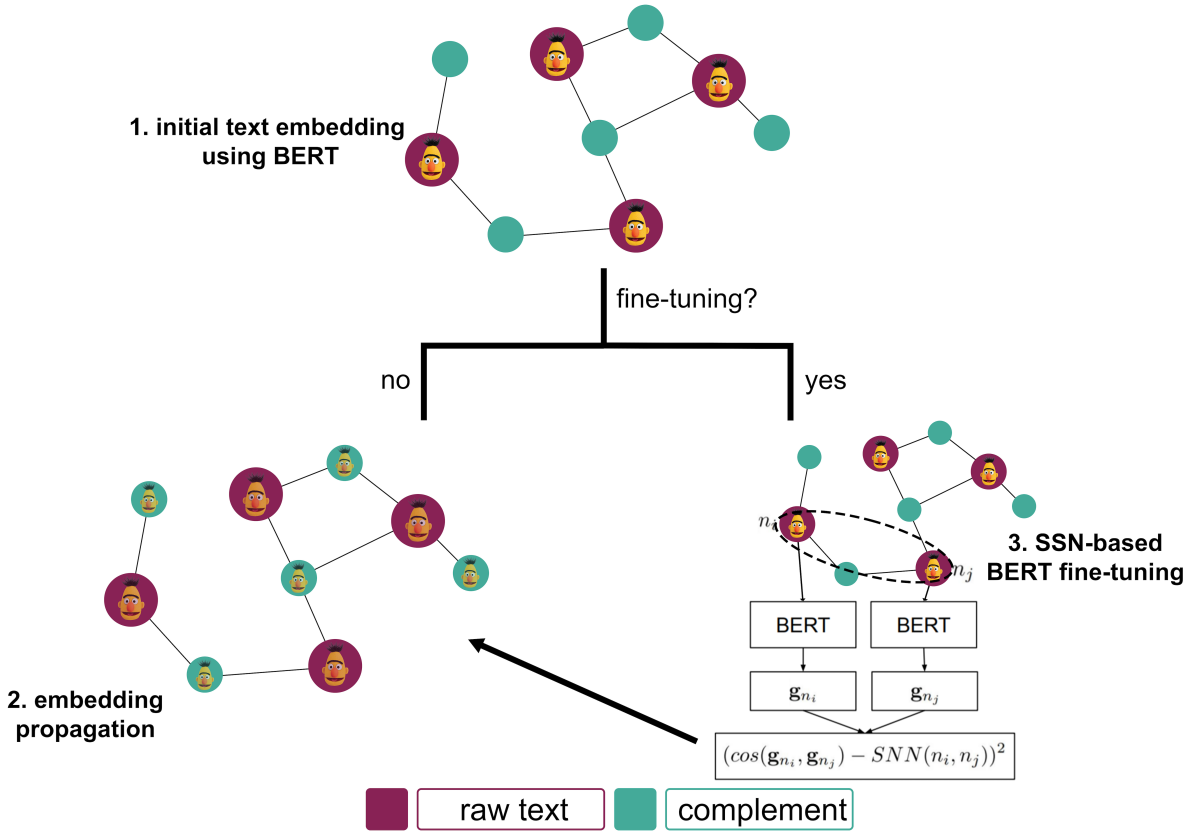


Figure 7 – Illustration of the pipeline of the proposed embedding propagation method, EPHEN. Source: authors.

3.2.1 Initial text embedding using BERT

The process of training and obtaining embeddings from the BERT model can be explored as defined in Equation 3.1,

$$p(t|c) = \frac{exp(\mathbf{h}_c^\top \mathbf{w}_t)}{\sum_{t'} exp(\mathbf{h}_c^\top \mathbf{w}_{t'})} \tag{3.1}$$

where $\mathbf{h}_c$ is a context embedding and $\mathbf{w}_t$ is a word embedding of the token t. BERT models are pre-trained in massive corpora and is fine-tuned for multilingual inputs, as presented in

(REIMERS; GUREVYCH, 2020). Generally BERT models fine-tuned in texts related to the application context perform better, but general knowledge methods can also be used when specific data is not available.

The start-up embedding for EPHEN is generated from raw texts $r_i = (t_1, ..., t_k)$, and outputted as $\mathbf{g}_r$ for a text r. Each text embedding is generated with the default behavior of Sentence-BERT (SBERT) (REIMERS; GUREVYCH, 2019) by taking the average of all token BERT embeddings $\mathbf{g}_d = \sum_{j=1}^{k} \frac{1}{k} \mathbf{w}_{t_j}$. In the heterogeneous event network, each raw text r is represented by a node $o \in O_r$ and directly associated with its respective initial embedding $\mathbf{g}_r$ computed from the SBERT model as shown in Figure 8. Although SBERT embeddings can represent node documents, other characteristics from the domain of the documents in the network, such as nodes of time and names of objects, people, or organizations, are also crucial for generalizing knowledge. With the initial embeddings generated for the raw text nodes O_r, EPHEN uses a regularization function to generate a final embedding that considers other complement nodes on the heterogeneous information network.

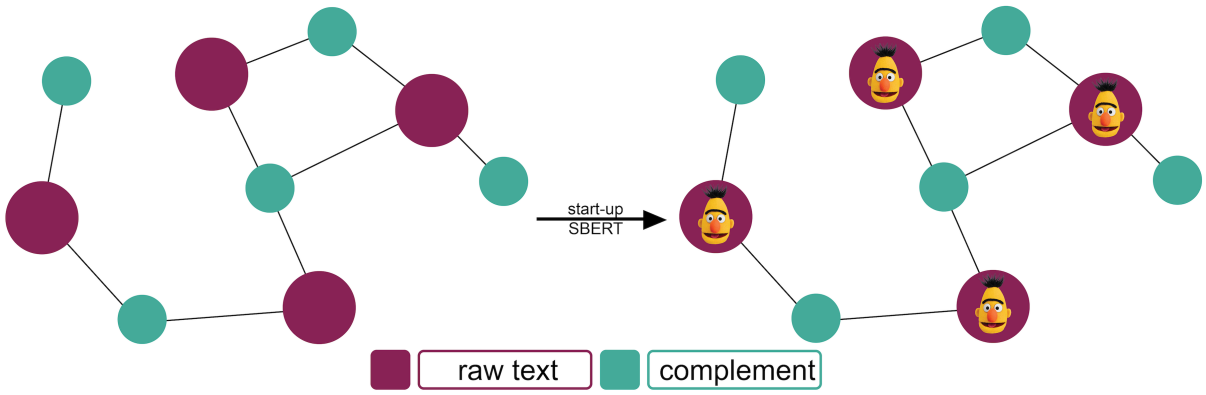


Figure 8 – A visualization of the start-up SBERT embedding insertion in the heterogeneous information network. Source: authors.

3.2.2 Embedding propagation

After the HINs have received their initial SBERT embedding, the regularization process can occur. Unlike Yang, Zhang and Han (2019) this master's dissertation proposes an embedding propagation method based on a regularization function (ZHOU *et al.*, 2004). This regularization is presented in Equation 3.2, where o_i and o_j are the neighboring nodes that must have similar embedding vectors $\mathbf{f}_{o_i}$ and $\mathbf{f}_{o_j}$. Moreover Ω is a distance equation and w_{o_i,o_j} are the weights of their relations. In the second part of the equation the start-up SBERT embedding $\mathbf{g}_{o_i} \in \mathbb{R}^d$ of raw text nodes O_r regulates the vector space according to a hyperparameter μ (where $\mu > 0$).

$$Q(\mathbf{F}) = \frac{1}{2} \sum_{o_i,o_j \in O} w_{o_i,o_j} \Omega(\mathbf{f}_{o_i}, \mathbf{f}_{o_j}) + \mu \sum_{o_i \in O_r} \Omega(\mathbf{f}_{o_i}, \mathbf{g}_{o_i}) \tag{3.2}$$

Although this equation can be used in an embedding propagation pipeline, it works better when extended (JI *et al.*, 2010). For this master's dissertation, the start-up embedding is outputted by a pre-trained SBERT model, and the dimension size d is determined by it. The Equation 3.3 defines the extensions and modifications applied to better accommodate the embedding propagation methods in HINs. This equation considers nodes of different types separately, and the results of the operations from a distance between embedding vectors are absolute values since this function was designed to propagate labels, and embedding vectors can accommodate negative values. Even with these modifications, the convergence of this regularization function has been proven (ZHOU *et al.*, 2004). It is minimized through quadratic operations or an iterative propagation process.

$$Q(\mathbf{F}) = \frac{1}{2} \sum_{o_r \in O_r} \sum_{o_c \in O_c} w_{o_r,o_c} \|\mathbf{f}_{o_r} - \mathbf{f}_{o_c}\|^2 + \mu \sum_{o_r \in O_r} \|\mathbf{f}_{o_r} - \mathbf{g}_{o_r}\|^2 \tag{3.3}$$

Figure 9 presents an example of EPHEN's execution. The first portion shows a heterogeneous event network before propagation, meaning that only event nodes have the start-up SBERT embedding. In the second portion, the network is shown after the propagation, meaning that all nodes have an adjusted final embedding but still belong to the initial SBERT vector space.

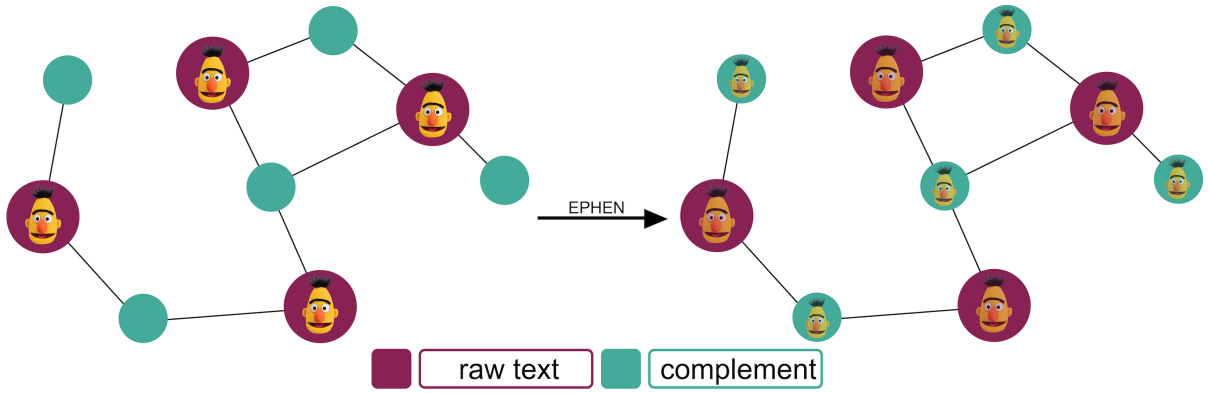
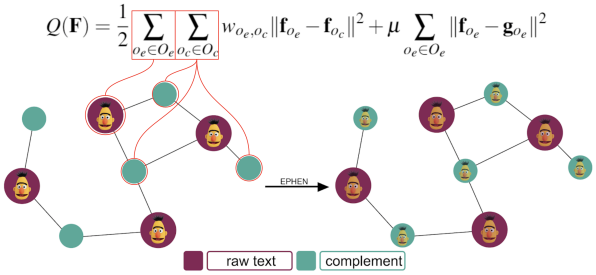
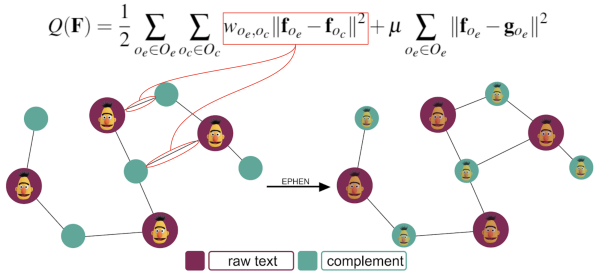


Figure 9 – A visualization of EPHEN's execution. Each node type has a different color, and the opaque BERT icon defines the start-up embedding. The final embeddings are defined by a translucent BERT icon, which shows they were adjusted by the network's topology and types of relations. Source: authors.

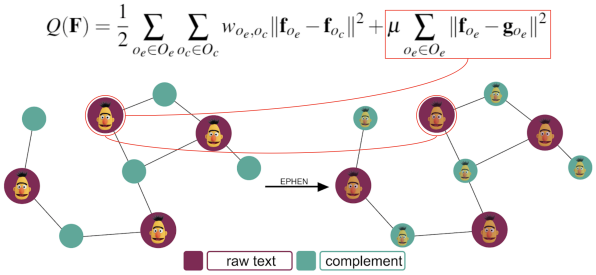
Figure 10 shows the embedding propagation steps highlighting what portions of the regularization it represents on the example. The modifications considering the difference for nodes with and without textual data are presented in Figure 10a. Figure 10b shows a modification to the original regularization is which the distances from neighbors are measured as absolute values to deal with embeddings instead of one-hot encoded labels. Figure 10c shows the hyper-parameter μ that whenever the value is closer to 1 the regularization tends to be more similar to the start-up embedding. And the first portion of the function is responsible for inserting the topology information in the final embedding as shown in Figure 10d.



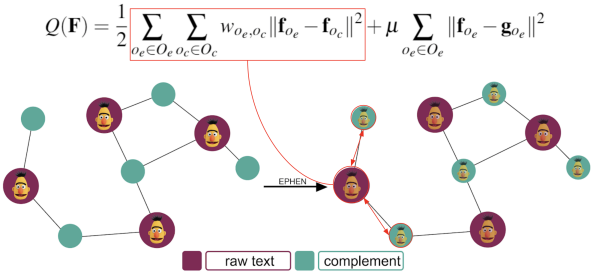
(a) The nodes that contain textual data are accounted separately from the ones that do not contain it.



(b) The distances between these types of nodes are accounted as absolute values since embeddings can contain negative values.



(c) The second portion, where the hyperparameter μ amounts for how much of the final embedding will tend towards the start-up embedding.



(d) The first portion where the neighborhood and distances account for the topology knowledge extracted from the network.

Figure 10 – Step by step demonstration of the regularization function used for embedding propagation. Source: authors.

3.2.3 SSN-based BERT fine-tuning

Since EPHEN adapts its regularization stage to any start-up embedding vector, it is possible to use both pre-trained and fine-tuned embedding models. Although BERT fine-tuning is a good option for obtaining better embeddings in a particular domain, it needs large quantities of labeled data. Since EPHEN is an unsupervised embedding propagation method, it will more often than not be applied to unlabeled data.

Thus it is essential to consider unlabeled data for fine-tuning as well. Conveniently the SBERT model first used as the start-up embedding for EPHEN has an architectural characteristic that can be leveraged for unsupervised fine-tuning. SBERT uses Siamese neural networks to aggregate tokens in sentences while generating a single output (REIMERS; GUREVYCH, 2019). Thanks to that, a fine-tuning method adjusts the BERT by approximating sentence similarity to node similarity.

Figure 11 shows an overview of the fine-tuning pipeline. The SBERT model is adjusted using the topological properties of the event network. Let n_i and n_j be two raw text nodes. They are connected with nodes representing complement characteristics to the raw text data like person or object names and others. Equation 3.4 defines the Shared Nearest Neighbors (*SNN*) measure, which calculates the similarity between raw text nodes from the network topology, measuring the number of shared complement nodes,

$$SNN(n_i, n_j) = \frac{\mathscr{S}(n_i) \cap \mathscr{S}(n_j)}{\mathscr{S}(n_i) \cup \mathscr{S}(n_j)} \tag{3.4}$$

where $\mathscr{S}(n_i)$ returns the set of neighboring nodes to n_i. The *SNN* measure varies in the range $[0, 1]$, meaning the more shared neighbors, the closer to 1 and the greater the topological similarity between the two nodes.

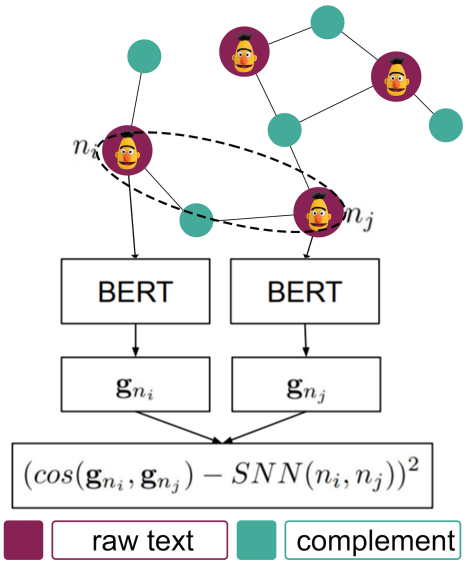


Figure 11 – Illustration of the fine-tuning using node similarity pipeline. Source: authors.

The Siamese neural networks use two different input vectors (e.g., initial SBERT event embeddings) to compute comparable output vectors (fine-tuned event embeddings). While the existing approaches use predefined manual scores between pairs of texts as similarity ground truth (which may be unfeasible in real-world applications), the final value is adjusted using the *SNN* measure as a target for fine-tuning. In this case, the embeddings are fine-tuned so that the cosine similarity of the raw texts $cos(\mathbf{g}_{n_i}, \mathbf{g}_{n_j})$, where $\mathbf{g}_{n_i}$ and $\mathbf{g}_{n_j}$ using the BERT embeddings of raw text nodes n_i and n_j respectively, is approximated by $SNN(n_i, n_j)$ — which represents topological properties of the heterogeneous information network. The loss function for the proposed fine-tuning is the *MSE* function (mean squared error), according to Equation 3.5,

$$MSE = \frac{1}{k} \sum_{i=1}^{k} (cos(\mathbf{g}_{n_i}, \mathbf{g}_{n_j}) - SNN(n_i, n_j))^2 \tag{3.5}$$

where k is the number of raw text pairs extracted from the heterogeneous information network for the BERT fine-tuning process. After the fine-tuning process, these are the start-up embeddings used in the embedding propagation function described in Equation 3.3.

3.2.4 Evaluation criteria

Since EPHEN is an unsupervised method that can be fine-tuned with unlabeled data, it needs to be evaluated in unsupervised tasks. So EPHEN is evaluated with graph completion tasks. Graph completion is derived from link prediction and can be understood as masking some known relationships in a network. After that, the model has to learn to predict these hidden relationships correctly (Figure 12). This master's dissertation evaluates graph completion separately on the desired type of relations. In order to evaluate only the quality of the generated embeddings, the predicted output will be a sorted list by the similarity between a pair of node embeddings.

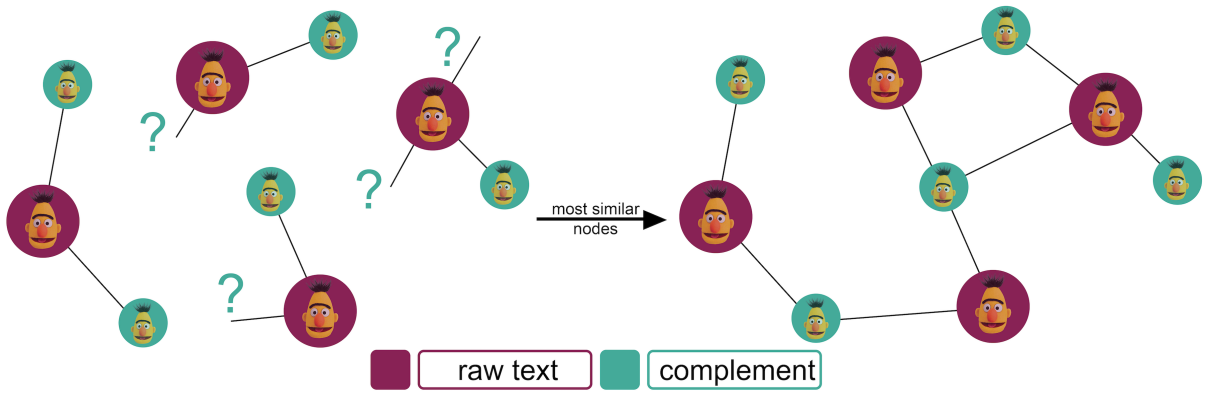


Figure 12 – Illustration of a graph completion task solved using node similarity on EPHEN's embeddings. Source: authors.

Within this master's dissertation, the default graph completion predictions will be obtained through a sorted list of the similarity between all possible pairs of relations from the desired type in a HIN. This behavior allows evaluation metrics for one correct prediction in a list of recommendations. More specifically, this master's dissertation will use evaluation metrics like MRR, $MRR@k$, and $Hits@k$ that were discussed in the subsection 2.1.3. Since these relations can be hidden randomly, the splits can be created using either train-[validation]-test split or k-fold cross-validation, which were also discussed in the subsection 2.1.3.

CHAPTER

4

EMBEDDING PROPAGATION OVER HETEROGENEOUS EVENT NETWORKS

This chapter presents the paper published in Carmo and Marcacini (2021). The source codes from the method and experiments are available at <https://github.com/PauloRVdC/ephen-experiments>. The paper authors were Paulo do Carmo and Ricardo Marcacini.

4.1 Initial remarks

Events can be defined as an action or a series of actions that occur at a specific time and place (ALLAN, 2012; CORDEIRO; GAMA, 2016; CHEN; LI, 2020). They are happening and evolving all the time. In this sense, social networks and news portals act as digital sensors for events happening around the world (DENG; RANGWALA; NING, 2019), allowing event analysis tasks. Different types of event analysis can allow real-time monitoring for application domains like the economy, epidemics, agribusiness, medicine, sentiment analysis, and many other social behavior studies (MARCACINI *et al.*, 2017).

Event analysis is the computational task for automatically identifying related events through text and other data such as timestamps, places, people, and entities involved (HAMBORG *et al.*, 2018; XUE *et al.*, 2019). In general, event analysis relies on unsupervised learning methods such as clustering (FLORENCE; NOGUEIRA; MARCACINI, 2017), and supervised learning such as classification (SANTOS; ROSSI; MARCACINI, 2017) and link prediction (RADINSKY; HORVITZ, 2013; NING *et al.*, 2019). Event analysis is challenging since different topics can be related at some level. For example, a country that changes its economic politics can change the behavior of consumers and the stock market and even trigger social acts (DENG; RANGWALA; NING, 2019).

Recently, heterogeneous networks have been used successfully for modeling large event datasets (SHI *et al.*, 2016). They model the different components of events as nodes (e.g.,

events, actors, locations, people, themes, and organizations), and network links express different relationships between these nodes. Thus, heterogeneous information networks (HIN) represent real-world objects through their connections among themselves.

Link prediction has a wide range of applications involving relationships between objects, in which the objective is to predict the probability of the link between two nodes. The most straightforward link prediction strategy measures the similarity between two nodes. The greater the value of the similarity function, the greater the probability of the link between nodes (YANG; LICHTENWALTER; CHAWLA, 2015; SHI *et al.*, 2016; KUMAR *et al.*, 2020). In this paper, we explore one of the ways to solve the link prediction problem for event analysis (LIBEN-NOWELL; KLEINBERG, 2007), which is to measure similarity between two nodes and determine the existence of a significant similarity considering events and their components. However, we must first define a unified feature space from heterogeneous data to calculate the similarity between nodes in heterogeneous networks.

Recently, network embeddings have been proposed as promising methods to learn features for nodes (WU *et al.*, 2020). Network embeddings methods map each node into a low-dimensional vector representing the network topology. Node type information (CHANG *et al.*, 2015; HUANG; MAMOULIS, 2017; SETTY; HOSE, 2018; CUI *et al.*, 2018; WU *et al.*, 2020). The recent literature has explored several methods to solve this task, including from random walks to deep learning methods (HAMILTON; YING; LESKOVEC, 2017; ZHANG *et al.*, 2019; WU *et al.*, 2020; HU *et al.*, 2020).

Existing network embeddings methods fail to meet two critical requirements to support link prediction in event networks. The first is to deal with different event components and metadata, in which event-type nodes have unstructured textual information. However, nodes representing location (e.g., latitude and longitude), time, person, and organizations are categorical or numeric features. The second is to deal with the inductive behavior of link prediction tasks, in which new nodes can be added to the heterogeneous network continuously. We argue that it is unfeasible to regenerate network embeddings for the entire network for each new event and its components. Thus, we raise the following question: *how to incrementally learn a unified feature space considering the textual information of event nodes with other non-textual node types of the event network*?

This paper presents a language model-based embedding propagation method for heterogeneous event networks, called Embedding Propagation over Heterogeneous Event Networks (EPHEN). While most of the existing network embedding methods mainly explore the network's topology, our method maps both: (i) textual information about events; and (ii) the complex relationships between events and their components to a low-dimensional vector space in order to use link prediction methods. Since events can be partially represented as texts, we can use neural language models, such as BERT (Bidirectional Encoder Representations from Transformers) (DEVLIN *et al.*, 2018), to compute meaningful representations. Moreover, neural language

models are pre-trained with billions of tokens, including news headlines. They have valuable general-purpose knowledge for event prediction. To our knowledge, we propose the first method to transfer textual embedding from pre-trained language models to heterogeneous event networks. In addition, we extend a regularization framework to adjust pre-trained language models' embeddings according to the topology of the event network.

We evaluate the proposed EPHEN method with other information network embeddings methods on real-world events datasets. We demonstrate the strengths of enrolling the general knowledge of a pre-trained neural language model into a multi-typed event information network for link prediction tasks through quantitative metrics and qualitative analysis. Furthermore, our experiments demonstrate that our EPHEN method generates competitive embeddings against state-of-the-art network embeddings, allowing dynamic and incremental updating as new events arise.

In this version of the paper used for the master's dissertation the related works section is omitted since it is contained within Section 2.3. It presented an introduction with concepts and motivation. It presents a proposal section with formulations and explanations. Moreover, it presents an experimental section with the used datasets, machines, achieved results, and discussions about them. Finally, it presents a conclusion wrapping up the contributions, limitations, and future work for this paper.

4.2 Embedding Propagation over Heterogeneous Event Networks

A heterogeneous event network is formally defined as a triple $N = (O, R, W)$, where O is the set of object nodes, R is the set of connections between objects, and W is the weight of those connections (SANTOS *et al.*, 2020). Heterogeneous event networks are information networks with the set $O > 1$, meaning they have more than one object node type. That allows the representation to adapt to a variety of real-world data and event analysis (Figure 13), in particular events represented by multiple components, such as places, time, names of people, and organizations. In this case, the set of objects is defined as $O = \{O_e \cup O_g \cup O_a \cup O_p \cup O_s\}$, where subset O_e represents event nodes, O_g represents geographical location nodes, O_a represents actor nodes (e.g. people or organizations), O_p represents theme nodes, and O_s represents temporal information nodes.

Our paper investigates network embedding through embedding propagation (YANG; ZHANG; HAN, 2019), where we can take advantage of initial event embeddings using a pre-trained neural language model. This version omits the method explanation section since the general problem for embedding propagation proposed in this paper is presented in Chapter 3. It is adapted for dealing with the specific characteristics extracted from an event dataset.

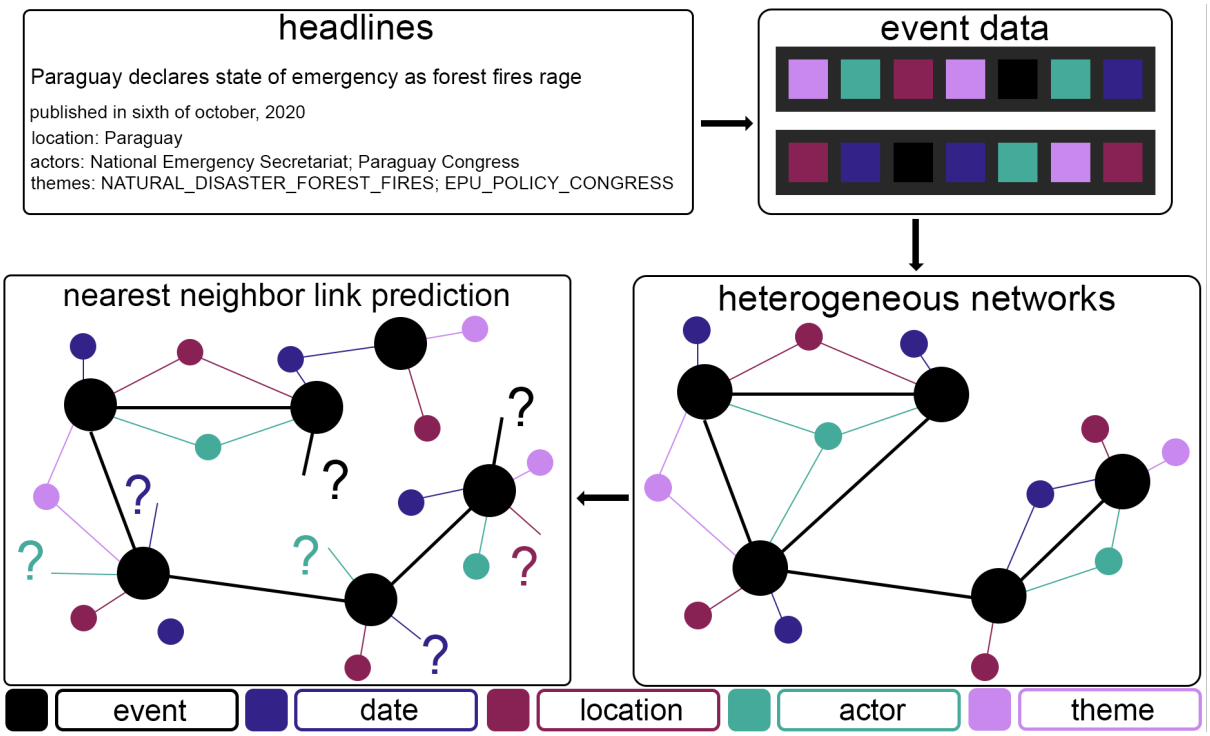


Figure 13 – An example of event analysis from heterogeneous networks. The network maintains complex relationships between the different event components. Network embeddings allow link prediction tasks and general queries to determine event predecessors of a target event. Source: (CARMO; MARCACINI, 2021).

4.3 Experiment evaluation

4.3.1 Datasets

To evaluate the proposed method under real-world scenarios, we extracted events from the GDELT Project [1] within 09/2019 and 10/2020. Table 1 presents an overview of the event networks used in the experimental evaluation, containing the total number of nodes, number of events, and the total number of nodes for each component. We use the DistilBERT-multilingual[2] pre-trained model to generate the initial event embeddings.

Table 1 – Overview of the heterogeneous event networks used throughout the experiments.

Network	#Nodes	#Events	#Dates	#Locations	#Actors	#Themes
#1	1272	161	43	141	343	584
#2	908	100	42	109	188	469

4.3.2 Baselines

We used the following baseline methods, which are explained in Chapter 2: DeepWalk, Node2Vec, Metapath2Vec, Struc2Vec, LINE, and GCN.

[1] Available at: <https://www.gdeltproject.org/>
[2] Available at: <https://github.com/UKPLab/sentence-transformers>

We chose these baselines since they represent shallower methods like EPHEN. Furthermore, GCN was used to represent inductive methods, but since the network does not contain labels, it learns embeddings according to event types. For the baselines, we use the parameters recommended by the authors in the respective original papers for each baseline method. Regarding the number of dimensions of the embeddings, we used 512 for all methods since it is the dimension used by the DistilBERT pre-trained model.

4.3.3 Evaluation Criteria

We evaluate EPHEN with the $MRR@k$ metric, which is explained in the Subsection 2.1.3.

EPHEN is configured to a $\mu = 0.85$ value and 15 iterations. In our experimental settings, three link prediction tasks were executed: event $\rightarrow$ event; event $\rightarrow$ location; and event $\rightarrow$ actor.

We also measured execution times for EPHEN and baseline methods. All experiments were executed in a machine with an AMD Ryzen 5 2600X; 32GB of RAM; an Nvidia Geforce GTX 1070; Ubuntu 21.04 as the OS; and a conda[3] environment for Python 3.8.10. It is important to reinforce that EPHEN is sequential, and DeepWalk and Node2Vec were executed as sequential methods in the CPU. In contrast, Metapath2Vec and Struc2Vec are parallel on the CPU, and LINE and GCN are parallel on GPU.

We also evaluated EPHEN's performance when new nodes were added after the first propagation, and new ones were generated upon the existing values since it maintains the same semantical space provided by the language model. In this scenario, we: (1) hide 40% of event nodes from a network; (2) propagate the embeddings with 15 iterations; (3) return 50% from the 40% hidden; propagate again with 5 iterations; (4) return the rest of the 50% from the %40 hidden; and (5) propagate for the last time with 5 iterations to obtain the final embeddings. Dynamic EPHEN's experiments were conducted in two tasks: event $\rightarrow$ location and event $\rightarrow$ actor link prediction tasks.

We compared t-SNE plots for EPHEN and three baseline methods: DeepWalk, Metapath2Vec, and GCN, for a visualization analysis. All t-SNE representations were constructed with $perplexity = 50.0$ and otherwise default settings.

All the results for link prediction scenarios were obtained after ten runs in random *train-test splits*. The heterogeneous networks and source codes are available at a GitHub repository[4].

4.3.4 Results and discussions

The first link prediction task defines an event $\rightarrow$ event scenario, which is a difficult task due to its correlation with event prediction and precursor recognition. Next, the event $\rightarrow$ location

[3] https://www.anaconda.com/

[4] Available at: <https://github.com/PauloRVdC/ephen-experiments>

link prediction task requires spatial abstraction for correlating multiple events. Finally, the event → actor link prediction task requires the model to generalize different words for describing a common actor like: 'President', 'Trump' and 'Government' might represent the same person in different events.

Table 2 presents the scores for three different levels *MRR@k* and for different amounts of edges removed before the link prediction evaluation. Our model has obtained the best results in all metrics configurations and percentages of splits, especially for Network #2. That indicates that pre-trained language models add valuable information to event analysis models. However, since our network builds the event → event links through text similarity, EPHEN's good performance was expected. On the other hand, the topology models like DeepWalk and Node2Vec were able to obtain second-best results, which indicates that the network topology also adds information to this event analysis scenario.

Table 2 – Average *MRR@k* score to the link prediction scenario event → event. The best results are bold.

	Network #1												**Network #2**											
	MRR@1				**MRR@3**				**MRR@5**				**MRR@1**				**MRR@3**				**MRR@5**			
	5%	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**
DeepWalk	**0.1**	**0.1**	0.05	0.03	0.1	0.1	0.05	0.05	0.1	0.1	0.09	0.07	0	0.1	0.08	0.04	0.03	0.13	0.08	0.05	0.03	0.13	0.08	0.05
Node2Vec	**0.1**	0	0	0	0.1	0.03	0.05	0.07	0.1	0.03	0.08	0.08	0.1	0.1	0.1	0.08	0.10	0.13	0.10	0.09	0.10	0.13	0.11	0.10
Metapath2Vec	0	0	0	0.03	0	0	0.03	0.06	0	0	0.06	0.06	0	0	0	0.02	0.03	0.03	0	0.02	0.03	0.03	0	0.03
Struc2Vec	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.01	0	0	0	0.01	0
LINE	0	0	0	0	0.05	0.05	0	0	0.05	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GCN	0	0	0	0	0	0.03	0	0.02	0.02	0.03	0	0.02	0	0	0	0	0	0	0	0	0	0.01	0	0
EPHEN	**0.1**	**0.1**	**0.1**	**0.07**	**0.17**	**0.17**	**0.15**	**0.13**	**0.21**	**0.21**	**0.20**	**0.16**	**0.20**	**0.15**	**0.15**	**0.14**	**0.23**	**0.20**	**0.20**	**0.19**	**0.25**	**0.21**	**0.21**	**0.20**

In Table 3 we observe the scores for event → location link prediction scenario. These nodes are components from events and are not directly related to the headline's text, and EPHEN was still capable of obtaining the best results overall. That shows us that the embedding propagation was able to insert network topology data within DistilBERT's embeddings. The scenario event → actor shows a similar behavior, as seen in Table 4. Once again, DeepWalk and Node2Vec's results were closest to EPHEN. Moreover, in this scenario, they were the closest out of the three. Since multiple names can represent the same actor, the more structured network topology extraction from the baselines can obtain more information.

Table 3 – Average *MRR@k* score to the link prediction scenario event → location. The best results are bold.

	Network #1												**Network #2**											
	MRR@1				**MRR@3**				**MRR@5**				**MRR@1**				**MRR@3**				**MRR@5**			
	5%	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**
DeepWalk	0.15	0.12	0.12	0.10	0.18	0.14	0.15	0.13	0.18	0.15	0.15	0.13	0.10	0.08	0.07	0.06	0.11	0.10	0.09	0.08	0.12	0.10	0.10	0.09
Node2Vec	0.15	0.12	0.11	0.09	0.16	0.14	0.13	0.12	0.17	0.15	0.14	0.12	0.08	0.08	0.07	0.06	0.09	0.09	0.09	0.07	0.09	0.09	0.09	0.08
Metapath2Vec	0.07	0.06	0.07	0.07	0.09	0.09	0.10	0.08	0.10	0.10	0.10	0.09	0.06	0.05	0.03	0.03	0.07	0.06	0.05	0.04	0.07	0.06	0.05	0.04
Struc2Vec	0.12	0.11	0.11	0.11	0.13	0.13	0.13	0.13	0.14	0.14	0.14	0.13	0.10	0.12	0.08	0.09	0.14	0.14	0.11	0.11	0.14	0.15	0.12	0.12
LINE	0	0.1	0	0	0.01	0.01	0.01	0.01	0.02	0.02	0.01	0.01	0	0	0	0	0	0	0	0	0	0.02	0.02	0.01
GCN	0.08	0.05	0.07	0.03	0.11	0.10	0.09	0.06	0.12	0.11	0.10	0.07	0.06	0.05	0.06	0.01	0.10	0.07	0.09	0.02	0.12	0.09	0.10	0.03
EPHEN	**0.30**	**0.25**	**0.27**	**0.24**	**0.37**	**0.34**	**0.35**	**0.33**	**0.40**	**0.37**	**0.38**	**0.36**	**0.14**	**0.15**	**0.13**	**0.12**	**0.22**	**0.22**	**0.21**	**0.19**	**0.25**	**0.24**	**0.23**	**0.22**

We also measured execution times for the link prediction tasks for each network embedding method. In Table 5 we present the time in seconds from obtaining embeddings to returning the predicted ordered list of correct links to that node. All the network embedding methods have the same nodes to predict links and use the same algorithm. GCN has the lowest execution time

Table 4 – Average *MRR@k* score to the link prediction scenario event → actor. The best results are bold.

	Network #1												**Network #2**											
	MRR@1				**MRR@3**				**MRR@5**				**MRR@1**				**MRR@3**				**MRR@5**			
	5%	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**
DeepWalk	0.19	0.17	0.16	0.14	0.21	0.20	0.19	0.18	0.22	0.20	0.19	0.18	0.16	0.16	0.15	0.14	0.19	0.19	0.18	0.18	0.20	0.20	0.19	0.19
Node2Vec	0.18	0.17	0.14	0.14	0.21	0.19	0.17	0.17	0.21	0.20	0.17	0.18	0.16	0.15	0.15	0.15	0.18	0.18	0.18	0.19	0.19	0.19	0.19	0.19
Metapath2Vec	0.03	0.04	0.02	0.02	0.04	0.05	0.04	0.03	0.04	0.05	0.05	0.03	0.05	0.07	0.05	0.06	0.07	0.09	0.08	0.08	0.08	0.10	0.09	0.09
Struc2Vec	0.01	0.02	0.01	0.02	0.03	0.04	0.03	0.03	0.03	0.05	0.03	0.04	0.05	0.04	0.05	0.05	0.06	0.06	0.06	0.07	0.07	0.07	0.08	0.08
LINE	0	0	0.01	0	0	0.01	0.02	0.01	0	0.01	0.02	0.02	0.02	0.02	0.01	0.01	0.03	0.02	0.02	0.02	0.03	0.02	0.02	0.03
GCN	0.08	0.05	0.02	0.02	0.11	0.06	0.04	0.04	0.13	0.07	0.05	0.04	0.06	0.03	0.04	0.04	0.10	0.07	0.08	0.06	0.11	0.09	0.09	0.08
EPHEN	**0.24**	**0.24**	**0.22**	**0.20**	**0.26**	**0.26**	**0.25**	**0.24**	**0.26**	**0.26**	**0.25**	**0.24**	**0.19**	**0.20**	**0.19**	**0.19**	**0.23**	**0.24**	**0.24**	**0.24**	**0.23**	**0.25**	**0.25**	**0.25**

in all tasks, followed by EPHEN and Metapath2Vec. It is important to reinforce that from these three best execution time methods, EPHEN is the only sequential method.

Table 5 – The average execution times and standard deviation for network embedding learning and link prediction tasks are in seconds. The lowest values are bold.

	Network #1			**Network #2**		
	event → event	**event → location**	**event → actor**	**event → event**	**event → location**	**event → actor**
DeepWalk	22 ± 0.12	24.58 ± **0.15**	25.99 ± 0.89	13.20 ± 0.12	16.12 ± 0.15	14.76 ± 0.23
Node2Vec	13.99 ± 0.19	16.80 ± 0.26	18.40 ± 0.80	9.18 ± 0.29	11.64 ± 0.11	10.71 ± 0.16
Metapath2Vec	4.21 ± 0.07	7.64 ± 0.17	9.06 ± 0.40	3.04 ± 0.11	5.36 ± 0.10	4.70 ± 0.09
Struc2Vec	196.22 ± 1.72	206.15 ± 3.60	199.39 ± 5.14	121.19 ± 2.48	125.26 ± 3.54	121.34 ± 3.33
LINE	191.09 ± 1.84	187.53 ± 3.63	192.37 ± 0.71	127.55 ± 0.63	129.70 ± 1.34	129.13 ± 0.93
GCN	**1.78** ± **0.04**	**5.60** ± 0.46	**6.62** ± **0.14**	**1.94** ± 0.15	**4.37** ± 0.04	**3.66** ± 0.13
EPHEN	2.68 ± 0.16	6 ± 0.18	7.09 ± 0.42	2.30 ± **0.04**	4.67 ± **0.03**	3.99 ± **0.08**

We compare the EPHEN offline results with the dynamic insertion experiment. The event → location and event → actor link prediction tasks allow us to evaluate the capacity of EPHEN's dynamic insertion in real-time event analysis.

In Table 6 we present the results with the average score and standard deviation for the location link prediction scenario. In this dynamic evaluation, we compare the results with embeddings for an entire network with EPHEN and the dynamic insertion scenario, where events were added to the network at different moments. The results show that the dynamic version of EPHEN obtains similar results to the embeddings obtained offline and regenerated for the entire event network. However, we have observed that there is generally an increase in the standard deviation of the link prediction score, indicating that the incremental version is less stable than the offline version.

Table 6 – Average *MRR@k* score and standard deviation to the link prediction scenario event → location, with dynamic insertion to the embedding propagation. The highest average score and lowest standard deviation values are bold.

	Network #1											
	MRR@1				**MRR@3**				**MRR@5**			
	5%	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**
EPHEN	**0.30** ± **0.09**	0.25 ± **0.05**	**0.27** ± **0.04**	0.24 ± **0.03**	**0.37** ± **0.06**	**0.34** ± **0.03**	**0.35** ± **0.03**	0.33 ± **0.03**	**0.40** ± **0.06**	**0.37** ± **0.04**	**0.38** ± **0.04**	0.36 ± **0.03**
Dynamic EPHEN	0.29 ± **0.09**	**0.26** ± 0.08	0.25 ± 0.05	**0.26** ± 0.06	0.34 ± 0.1	**0.34** ± 0.1	**0.35** ± 0.07	**0.34** ± 0.07	0.37 ± 0.1	**0.37** ± 0.1	0.37 ± 0.07	**0.37** ± 0.06

Analogous results were obtained in the link prediction task between events and actors, as described in Table 7. In this case, different names for the same actor might appear in different moments and events of the network, and they might be positioned accordingly to the previous

state of the network. On the other hand, the scenario we built is the worst case for the dynamic EPHEN since events were hidden randomly.

Table 7 – Average *MRR@k* score and standard deviation to the link prediction scenario event → actor, with dynamic insertion to the embedding propagation. The highest average score and lowest standard deviation values are bold.

	Network #1											
	MRR@1				MRR@3				MRR@5			
	5%	10%	15%	20%	5%	10%	15%	20%	5%	10%	15%	20%
EPHEN	0.24 ± **0.08**	**0.24** ± **0.06**	**0.22** ± **0.04**	**0.20** ± **0.04**	0.26 ± **0.08**	**0.26** ± **0.06**	**0.25** ± **0.04**	**0.24** ± **0.04**	0.26 ± **0.08**	**0.26** ± **0.06**	**0.25** ± **0.04**	**0.24** ± **0.04**
Dynamic EPHEN	**0.26** ± 0.13	0.19 ± 0.09	0.18 ± 0.08	0.16 ± 0.07	**0.29** ± 0.14	0.21 ± 0.1	0.22 ± 0.07	0.20 ± 0.07	**0.30** ± 0.14	0.22 ± 0.1	0.22 ± 0.07	0.21 ± 0.07

Figure 14 shows t-SNE plots for EPHEN and three other baselines: DeepWalk, Metapath2Vec, and GCN. Regarding network #1 and the link prediction tasks, we can observe the plots in two aspects. First, we measure the plot quality concerning the network topology. Then, we restored the nearest point from each point representing an event node and analyzed if it corresponded to a link on the network. Then we averaged the correct *event* → [*component*] pair for each network embedding and used it as a quantitative metric for the plot. As a result, we can observe on the plot that EPHEN and DeepWalk create smaller communities with events and components, while Metapath2Vec and GCN generate embeddings by clustering node types. This behavior helps to explain the good performance from EPHEN and DeepWalk in the link prediction tasks as the nodes connected in the network are close, even with the t-SNE dimension reduction.

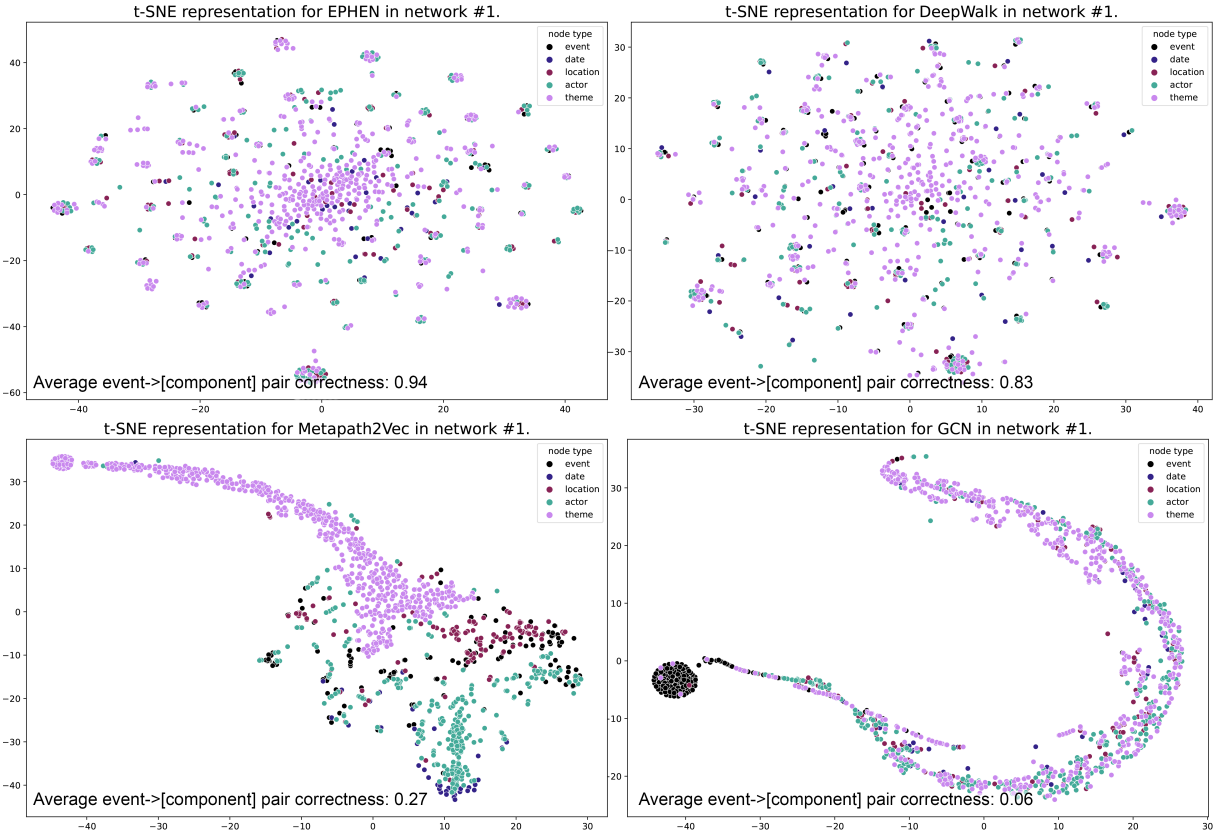


Figure 14 – t-SNE plots for EPHEN and three other baselines representing different network embedding approaches. Source: (CARMO; MARCACINI, 2021).

In general, these experiments show that EPHEN is competitive both in the performance of link prediction tasks and computational complexity. Specifically, in our heterogeneous star networks from GDELT data, EPHEN's embedding propagation performed best in every task. Furthermore, EPHEN has the second best execution, and in contrast to the best execution times achieved by GCN, EPHEN is a sequential method.

4.4 Concluding remarks

This paper introduces an embedding propagation method that transfers knowledge from pre-trained language models into heterogeneous events networks. Considering a more conceptual aspect, we adjust the semantic space of a neural language model according to the topology of a heterogeneous event network. In practice, our method takes advantage of dozens of pre-trained language models that are currently published and incorporates them into network event analysis. We show that this strategy is simple and effective, achieving competitive results in different link prediction tasks and naturally allowing the incremental update of embeddings as new events arise.

An analysis of the experimental results revealed that our proposal is competitive compared to the Deepwalk and Node2vec methods, which mainly explore the network topology. Based on these results, we plan to explore different importance levels for each relationship between events and their components, thereby incorporating more network topology information during the embedding propagation. Our experiments did not explore Deep Learning based methods like HetGNN, HGT, and GraphSAGE. Those methods require more computational resources and hyperparameter tuning than "shallow" strategies like DeepWalk, or regularization approaches like EPHEN.

Other directions for future work involve investigating EPHEN's performance against Deep Learning based methods for network embedding. We also pretend to extend EPHEN's regularization method for considering different weights to different types of nodes and evaluating its performance in different networks, tasks, and through different performance metrics. Finally, we reinforce that all source code, networks, and extended versions of the experiments are available at <https://github.com/PauloRVdC/ephen-experiments>.

CHAPTER

5

COMMODITIES TREND PREDICTION ON HETEROGENEOUS INFORMATION NETWORKS

This chapter presents a paper accepted in a periodic that is the extension of a conference paper. The original paper was published in Carmo, Filho and Marcacini (2021) and its extension was accepted but not yet published in the Journal of Information and Data Management (JIDM). The first paper method and experiments source codes are available at <https://github.com/PauloRVdC/tphin-experiments>. The extended paper presented here was chosen since it accommodates the method and original experiments while adding a fine-tuning pipeline and its experiments. The source codes from the extended paper, which contains the fine-tuning pipeline, are available at <https://github.com/PauloRVdC/TRENCHANT>. Both papers authors were Paulo do Carmo, Ivan José dos Reis Filho and Ricardo Marcacini.

5.1 Initial remarks

Events can be defined as an action or a series of actions that occur at a specific time and place (ALLAN, 2012; CORDEIRO; GAMA, 2016; CHEN; LI, 2020). Different types of event analysis allow real-time monitoring for application domains like the economy, agribusiness, epidemics, medicine, sentiment analysis, and many other social behavior studies (MARCACINI *et al.*, 2017). Agribusiness events are challenging to predict since they depend on (i) climate changes, (ii) historical data from the market, (iii) supply and demand, (iv) aggregate demand, and (v) politics. Climate changes are predictable through meteorologic data. Market history, as well as supply and demand data, are obtained from temporal series. However, aggregate demand and politics are also available in text data from the news or social networks. Since textual data is made for humans, text mining is a challenging task that requires multiple steps of processing (VENTER; STRYDOM; GROVÉ, 2013).

Event analysis is the computational task for automatically identifying related events through text and other data such as timestamps, places, people, and entities involved (HAMBORG *et al.*, 2018; XUE *et al.*, 2019). The data extracted comes in 5W1H model which is described as (CHEN; LI, 2020): **what** happened; **when** it happened; **where** it happened; **who** is involved; **why** it happened; and **how** it happened. Thus, these techniques can be used to explore useful agribusiness data for trend prediction (RADINSKY; HORVITZ, 2013; NING *et al.*, 2019), since they can be modeled as events. Event analysis is challenging since different topics can be related at some level. For example, the corn value can drop due to new exporting politics that lower international demand, and if identified earlier, a producer might be able to close a better domestic deal (FILHO *et al.*, 2020).

Recently, heterogeneous information networks (HINs) have been used successfully for modeling large event datasets (SHI *et al.*, 2016) since they model different components from events as nodes (e.g., when, where, who, why, and how), and network links express different relationships between these nodes. Thus, we can create a HIN representing each event by itself and the 5W1H. The use of HINs also allows network completion tasks. Network completion tasks derive from link prediction. It is the task of estimating the connection between two nodes based on observed links and node features. The main difference from link prediction is that the method only tries to connect nodes of specific types in network completion. It can also be defined as a simple binary classification problem (SHI *et al.*, 2016). Network completion can also be modeled as a multi-class classification problem whenever a set of node types can be used as labels. Network completion as a classification uses node features as data. There are many techniques for obtaining node features from a network.

Network embeddings methods map each node into a low dimensional vector that represents the network topology, and node type information (CHANG *et al.*, 2015; HUANG; MAMOULIS, 2017; SETTY; HOSE, 2018; CUI *et al.*, 2018; WU *et al.*, 2020). The recent literature has explored several methods to solve this task, from random walks within the entire network to deep learning methods (WU *et al.*, 2020). However, since events can be partially represented as texts, such as news headlines or social networks posts, we can use neural language models, such as the Bidirectional Encoder Representations from Transformers (BERT) (DEVLIN *et al.*, 2018), to compute meaningful representations. Neural language models are trained over large textual datasets, including news headlines, so they have valuable general-purpose knowledge for event prediction. However, using just text embedding is not recommended since it ignores essential component information (CHEN; LI, 2020), which may lead to deeper connections between events, like location and actors involved.

This paper presents the TrEnd pRediction on heteRogeneous InFormatIon nEtwoRks (TRENCHANT), a language model-based embedding propagation method for heterogeneous event networks. While most existing network embedding methods mainly explore the network's topology, our method maps both (i) textual information about events; (ii) the complex rela-

tionships between events and their components to a low-dimensional vector space; and (iii) a fine-tuning strategy that leverages topological properties from the heterogeneous information network to enrich the textual embeddings. Existing methods of network embeddings are offline, which requires repeating the entire process when new nodes are added to the network. Some pipelines may include different techniques for dynamic insertions (DENG; RANGWALA; NING, 2019; DENG; RANGWALA; NING, 2020), but they must be modeled apart from the network embedding. On the other hand, our proposed method is naturally incremental. It uses an initial neural language model embedding, pre-trained and fine-tuned respectively, to maintain a fixed vector space propagated to the entire network.

A preliminary version of our method was proposed in Carmo, Filho and Marcacini (2021). In this extended version, we incorporate the BERT model fine-tuning strategy using topological properties of the event network, as well as a more robust experimental analysis. Our main contributions are threefold:

- First, we propose a regularization framework for embedding propagation from a pre-trained BERT language model to all nodes of a heterogeneous event network. Thus, all network nodes are mapped to the same embedding space, allowing to compute similarities between textual and non-textual nodes of an event dataset.

- Second, we introduce a fine-tuning method of the BERT model from the topological properties of the event network. Although the proposed method can handle general-purpose pre-trained BERT models (TRENCHANT method), we discuss and evaluate scenarios in which fine-tuning the initial embeddings before propagation in the heterogeneous network yields promising results (FT-TRENCHANT method).

- Finally, we model the trend prediction problem as a classification task from the final embeddings extracted from an event network. That is a practical solution that allows different well-known classifiers, such as Long Short-Term Memory (LSTM) neural networks.

We evaluate the proposed methods with other information network embeddings methods on an agribusiness news dataset. We consider corn and soybean prices for trend prediction. We also demonstrate how enrolling the general knowledge of a pre-trained and fine-tuned neural language model into a heterogeneous information network embedding method performs in trend prediction scenarios. We show that using TRENCHANT and FT-TRENCHANT proposed methods is competitive with state-of-art network embeddings algorithms. Moreover, our proposal performs network embedding incrementally, allowing the insertion of new nodes in the same semantic space without rebuilding the entire network embedding.

5.2 Related Work

Commodities trend prediction is usually based on time series analysis techniques like Integrated Autoregressive Moving Average (ARIMA, (DAREKAR; REDDY, 2017)) and Integrated Seasonal Autoregressive Moving Average (SARIMA (ADANACIOGLU; YERCAN *et al.*, 2012)). With the advance in text mining techniques, some works began to combine these text features with time series data for stock and commodity prediction (WANG *et al.*, 2019; CHEN; CHEN; CHIU, 2016). In dos2020forecasting the authors combine text and time-series data. The text was obtained from a bag-of-words model and concatenated with a decision tree model based on time series data. They evaluate the models by comparing the results from time series and the proposed model in a Support Vector Regression model for commodity price prediction.

The remainder of this section is omitted since it is contained in Section 2.3.

5.3 Trend prediction on heterogeneous information networks

5.3.1 Event Modeling with Heterogeneous Networks

Events extracted from news data act as digital sensors, as they are published around the time they happened and contain information. This information can be extracted directly from the text and its metadata (HAMBORG *et al.*, 2018). We chose to extract a 4W1H variation without the *when* characteristics since it is not as good data as the publication date in this domain. We also consider metadata from commodities prices to generate the trend indicators of the temporal series. The trend labels are calculated according to the $[week|month]/year$ period and commodity they represent. With the news features extracted and trend labels calculated, we can model different heterogeneous networks to each combination (Figure 15).

A heterogeneous event network is formally defined as a triple $N = (O, R, W)$, where O is the set of object nodes, R is the set of connections between objects, and W is the weight of those connections (SANTOS *et al.*, 2020). Heterogeneous event networks are information networks with the set $O > 1$, meaning they have more than one object node type. That allows the representation to adapt to various real-world data and event analysis, in particular events represented by multiple components, such as places, time, names of people, and organizations. In this case, the set of objects is defined as $O = \{O_e \cup O_d \cup O_w \cup O_l \cup O_a \cup O_y \cup O_h \cup O_t\}$, where the subset O_e represents event nodes, O_d represents date nodes, O_w represents what nodes, O_l represents location nodes, O_a represents actors nodes (e.g. people or organizations), O_y represents why nodes, O_h represents how nodes and O_t represents trend nodes (our desired labels).

This version omits the embedding propagation and fine-tuning methods explanation subsections since they were presented in Chapter 3. They were adapted to address the specific

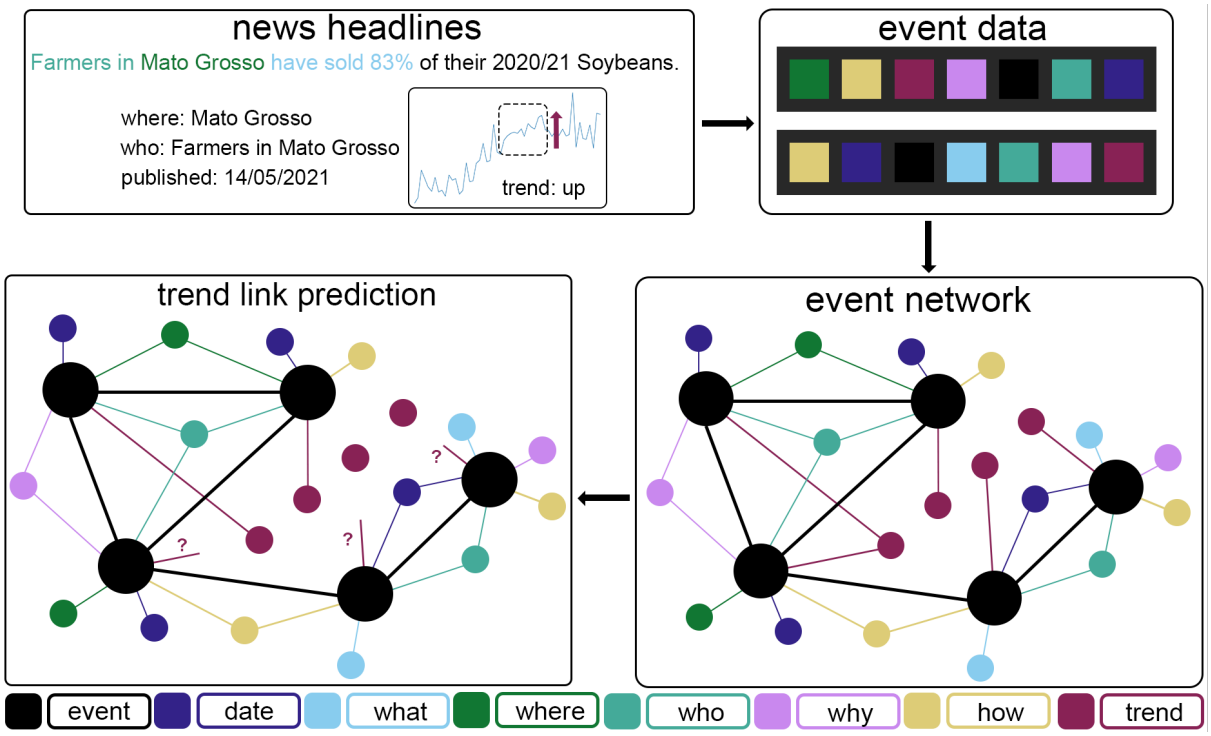


Figure 15 – Visual representation of the proposal. The event components are extracted from news headlines and metadata. The trend symbolizes the price trend of a commodity at the end of the period the news was published. Source: (CARMO; FILHO; MARCACINI, 2021; CARMO; FILHO; MARCACINI, 2022).

characteristics modeled in the proposed HIN. Instead, the following subsection presents the trend prediction pipeline, which is unique for this paper.

5.3.2 Trend Prediction

We must apply the data collected to a machine learning algorithm with the network embeddings calculated. In this paper, we have chosen a long-short term memory (LSTM) for supervised multi-class classification with the network embedding data. The LSTM we used accommodates memory units. The memory units are composed of three smaller units (u, c_{in}, c_{out}). The units u (Equation 5.1 (i)) apply a weighted sum, where w represents the weight, for each value y, and its result is outputted through an activation function. These units are limited by control units c_{in} (Equation 5.1 (ii)) and c_{out} (Equation 5.1 (iii)) that control when the iteration t will go forward or keep recurring. The initial value y inserted to the LSTM comes from the propagated embeddings f and changes throughout training.

$$\begin{aligned} (i) \quad & {u_u}^{(t)} = \textstyle\sum_u w_u y^{u(t-1)} \\ (ii) \quad & {c_{in}}^{(t)} = \textstyle\sum_u w_{in_j u} y^{u(t-1)} \\ (iii) \quad & {c_{out}}^{(t)} = \textstyle\sum_u w_{out_j u} y^{u(t-1)} \end{aligned} \tag{5.1}$$

These units must learn the correct weights to predict trends from an embedding vector.

Thus the LSTM is trained by a process called back-propagation. A back-propagation optimizer uses a gradient (e.g. Equation 5.2) that calculates the error E between the actual label from a training to the time step $T^{'}, T$ and generates a Δ to update each weight w_{ij} from the LSTM network.

$$\Delta w^{E^{total}}(T^{'},T) = \sum_{t=T^{i}+1}^{T} \Delta w^{E(t)} \rightarrow \Delta w_{ij} = -\alpha \frac{\partial E^{total}(T^{'},T)}{\partial w_{ij}} \tag{5.2}$$

The LSTM used in this proposal has: (i) a dense layer with as many units as the input (512) that uses a *relu* activation function (Equation 5.3), allowing the optimization process to turn units off when necessary; (ii) we mini-batch experiments and determined 64 memory units were the optimal number for these experiments; (iii) the last layer calculates the probabilities of each class since it uses a SoftMax (GOODFELLOW; BENGIO; COURVILLE, 2016) activation function; and (iv) it uses the Adam optimizer during training. With this LSTM, we can leverage the data represented by the embeddings in a neural classification, allowing non-supervised methods to compete with graph neural networks.

$$f(u) = \max(0,u) \tag{5.3}$$

5.4 Experimental evaluation

5.4.1 Datasets

We use a dataset related to agribusiness news to evaluate the trend prediction. It contains news related to the corn and soybean commodities extracted from Soybean & Corn Advisor[1] and the historical prices from the Centro de Estudos Avançados em Economia Aplicada (CEPEA)[2]. This dataset allows us to extract the events and components from the news text. We calculate Trends' classes in four ways, generating four networks as shown in Table 8. Each network has the same events and components but generates trends' labels nodes by different commodities and time windows. In addition, we use the DistilBERT-Multilingual[3] model to generate the initial embedding for each event headline text.

Table 8 – Overview of heterogeneous event networks used in the experimental evaluation.

Network	#Nodes	#Events	#Dates	#Whats	#Wheres	#Whos	#Whys	#Hows	Time window	Commodity
#1	4348	2322	380	48	115	9	244	1226	weekly	corn
#2	4348	2322	380	48	115	9	244	1226	weekly	soybean
#3	4056	2322	89	48	115	9	244	1226	monthly	corn
#4	4056	2322	89	48	115	9	244	1226	monthly	soybean

[1] Available at: <http://soybeansandcorn.com>
[2] Available at: <https://www.cepea.org.br>
[3] Available at: <https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v2>

5.4.2 Evaluation criteria and experiment setup

We use traditional metrics to evaluate the multi-class classification scenario: macro precision, macro recall, and macro $F1$. Another important metric to evaluate these methods is the $F1$ for the *big_down* and *big_up* classes. We consider it essential because these classes indicate a critical market tendency transition. It is important to know the performance of the methods on them. The evaluation scenarios consist of two windows for obtaining trends and two splits for predicting them on all four networks. We also evaluate average execution times. In Table 9 we present the training scenarios statistics for train and test size, according to the commodity and number of weeks or months used to generate the labels.

Table 9 – Overview of scenario configurations with train and test sizes.

Scenario	Time window	Commodity	#Weeks	#Months	Train size	Test size
#1	weekly	corn	24	-	2196	126
#2	weekly	corn	48	-	2037	285
#3	weekly	soybean	24	-	2196	126
#4	weekly	soybean	48	-	2037	285
#5	monthly	corn	-	6	2203	119
#6	monthly	corn	-	12	2029	293
#7	monthly	soybean	-	6	2203	119
#8	monthly	soybean	-	12	2029	293

With the experiment scenarios displayed, we also need to account for the class imbalance between all four labels (*big_down*, *down*, *up*, and *big_up*). Table 10 presents all class distributions for each scenario train/test split.

Table 10 – Overview of class balance for each scenario on $train/test$ splits.

Scenario	$big\_down(train/test)$	$down(train/test)$	$up(train/test)$	$big\_up(train/test)$
#1	97/5	999/57	977/57	123/7
#2	83/19	964/92	887/147	103/27
#3	189/12	918/63	898/41	191/10
#4	155/46	888/93	838/101	156/45
#5	28/0	1006/38	1093/81	76/0
#6	28/0	973/71	952/222	76/0
#7	26/23	1096/15	1000/61	81/20
#8	26/23	1096/15	846/215	61/40

We compare TRENCHANT and FT-TRENCHANT with state-of-art network embeddings methods: DeepWalk, Node2Vec, Metapath2Vec, Struc2Vec, LINE, and GCN. We use the parameters recommended by the authors in the respective original papers for each baseline method. Regarding the number of dimensions of the embeddings, we used 512 for all methods since it is the dimension used by the DistilBERT language model. All experimental data, source code, and networks are available at the GitHub repository <https://github.com/PauloRVdC/TRENCHANT>.

5.4.3 Results and Discussion

Considering the experiment setup explained previously, we will present results for each scenario, pointing out relevant overall and specific class performance results. We will also

evaluate the models considering stability through box plot variations on scenario results.

In Table 11 we present the results for the odd number scenarios (#1, #3, #5 and, #7), representing the shorter time splits. We can see TRENCHANT and FT-TRENCHANT are the best performers on most scenarios and metrics. More specifically, FT-TRENCHANT is best in both scenarios constructed with soybean trends. At the same time, TRENCHANT has the best $F1$ and better recall for scenario #1, which is the shorter time split for corn trends. We can also observe that TRENCHANT had worst performance than baseline methods on scenario #3, and FT-TRENCHANT managed to beat them with some margin. These results show that our fine-tuning pipeline adds some knowledge to the representation. However, it also shows that our fine-tuning pipeline does not help all scenarios since performance has decayed in all corn scenarios.

Table 11 – Trend prediction performance for scenarios #1, #3, #5 and, #7 on three metrics (macro $F1$, macro precision (pre) and, macro recall (rcl)). The highest scores are in bold.

	Scenario #1			**Scenario #3**			**Scenario #5**			**Scenario #7**		
	$F1$	**pre**	**rcl**	$F1$	**pre**	**rcl**	$F1$	**pre**	**rcl**	$F1$	**pre**	**rcl**
DeepWalk	0.24	0.24	0.26	0.23	0.23	0.26	**0.48**	**0.48**	**0.48**	0.13	0.12	0.21
Node2Vec	0.27	0.30	0.27	0.23	0.23	0.25	0.24	0.25	0.24	0.18	0.22	0.25
Metapath2Vec	0.22	**0.32**	0.27	0.20	0.28	0.26	0.11	0.29	0.08	0.15	0.21	0.23
Struc2Vec	0.17	0.22	0.22	0.20	0.23	0.23	0.18	0.29	0.14	0.19	0.21	0.27
LINE	0.24	0.25	0.24	0.25	0.26	0.26	0.31	0.32	0.31	0.18	0.21	0.25
GCN	0.27	**0.32**	**0.30**	0.21	0.22	0.23	0.39	0.42	0.45	0.14	0.12	0.24
TRENCHANT	**0.29**	0.30	**0.30**	0.21	0.22	0.24	0.23	0.24	0.23	0.21	0.30	0.24
FT-TRENCHANT	0.21	0.22	0.21	**0.33**	**0.34**	**0.34**	0.26	0.27	0.26	**0.25**	**0.31**	**0.35**

In Table 12 we present the results for the even number scenarios (#2, #4, #6 and, #8), representing the longer time splits. Compared to the shorter time splits, TRENCHANT and FT-TRENCHANT have lacked some performance, going from being the best in most scenarios to having the best $F1$ and precision on scenario #8. Nevertheless, this shows us that the fine-tuning process adds some knowledge that helps discriminate trends for the soybean commodity. We can also see that TRENCHANT is the second-best on scenario #2, which aligns with its good performance on scenario #1.

Table 12 – Trend prediction performance for scenarios #2, #4, #6 and, #8 on three metrics (macro $F1$, macro precision (pre) and, macro recall (rcl)). The highest scores are in bold.

	Scenario #2			**Scenario #4**			**Scenario #6**			**Scenario #8**		
	$F1$	**pre**	**rcl**	$F1$	**pre**	**rcl**	$F1$	**pre**	**rcl**	$F1$	**pre**	**rcl**
DeepWalk	0.24	0.26	0.27	0.20	0.27	0.24	0.36	**0.40**	0.39	0.16	0.20	0.24
Node2Vec	0.24	0.25	0.26	0.22	0.23	0.26	0.24	0.27	0.26	0.17	0.24	0.25
Metapath2Vec	0.19	0.25	0.20	**0.30**	**0.35**	**0.32**	0.10	0.28	0.07	0.13	0.19	0.24
Struc2Vec	0.20	0.23	0.22	0.22	0.24	0.24	0.16	0.28	0.12	0.17	0.22	0.22
LINE	0.24	0.26	0.25	0.22	0.23	0.24	0.25	0.28	0.27	0.17	0.26	0.25
GCN	**0.26**	**0.31**	**0.31**	0.19	0.17	0.24	**0.37**	0.38	**0.40**	0.16	0.24	**0.32**
TRENCHANT	0.25	0.26	0.27	0.19	0.18	0.20	0.22	0.25	0.23	0.18	0.38	0.28
FT-TRENCHANT	0.23	0.23	0.24	0.20	0.20	0.22	0.20	0.23	0.20	**0.20**	**0.39**	0.25

In Table 13 we present the results specific for the critical classes (*big_down* and *big_up*). We can observe that even though TRENCHANT and FT-TRENCHANT only achieve the best

results on scenarios #1 and #3, respectively, the overall performance guarantees a result throughout all scenarios. Scenarios #5 and #6 did not present any results for the *big_down* and *big_up* classes. Another takeaway from these results is that Metapath2Vec achieves the best performance in most scenarios, showing us that the meta-paths allow for good coverage of unbalanced class features.

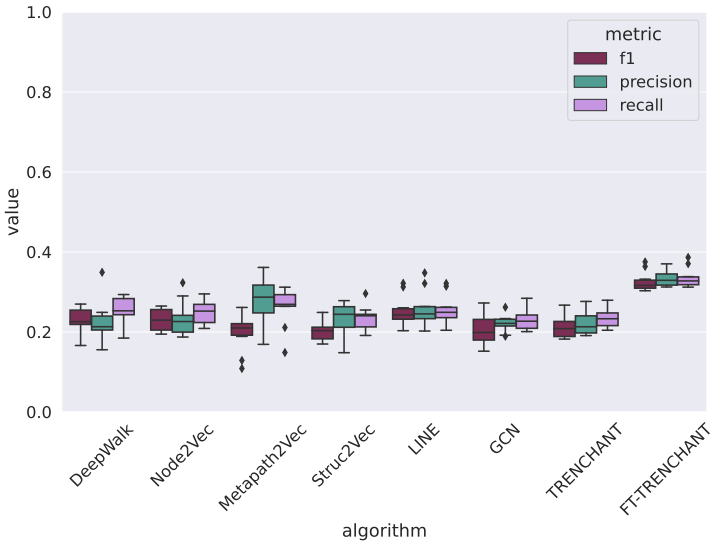
Table 13 – Trend prediction performance for all scenarios on two metrics (*big_down* $F1$ (bd), *big_up* $F1$, (bu)). The highest scores are in bold.

	Scenario #1		**Scenario #2**		**Scenario #3**		**Scenario #4**		**Scenario #5**		**Scenario #6**		**Scenario #7**		**Scenario #8**	
	(bd)	**(bu)**	**(bd)**	**(bu)**	**(bd)**	**(bu)**	**(bd)**	**(bu)**	**(bd)**	**(bu)**	**(bd)**	**(bu)**	**(bd)**	**(bu)**	**(bd)**	**(bu)**
DeepWalk	0.00	0.00	0.01	0.01	0.01	0.00	0.03	0.03	-	-	-	-	0.00	0.00	0.00	0.00
Node2Vec	**0.07**	0.02	0.02	0.01	0.03	0.03	0.04	0.02	-	-	-	-	0.00	0.05	0.01	0.03
Metapath2Vec	**0.07**	0.00	0.01	**0.12**	0.07	0.10	**0.29**	**0.25**	-	-	-	-	**0.28**	0.16	**0.27**	0.14
Struc2Vec	0.04	0.10	**0.08**	**0.12**	0.09	**0.13**	0.16	0.20	-	-	-	-	0.05	**0.21**	0.04	**0.20**
LINE	0.00	0.09	0.02	0.07	0.07	0.05	0.09	0.07	-	-	-	-	0.01	0.04	0.01	0.05
GCN	0.00	0.00	0.00	0.02	0.00	0.00	0.01	0.00	-	-	-	-	0.00	0.00	0.00	0.00
TRENCHANT	**0.07**	**0.12**	0.00	0.05	0.06	0.04	0.06	0.02	-	-	-	-	0.02	0.09	0.04	0.07
FT-TRENCHANT	0.00	0.02	0.00	0.07	**0.23**	0.01	0.03	0.02	-	-	-	-	0.04	0.09	0.07	0.10

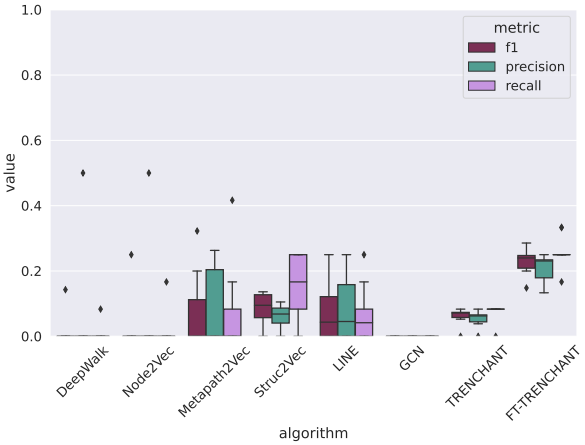
In Figure 16 we present five box plots from scenario #3 that showcase the stability of methods through five variations of the $F1$, precision, and recall metrics. The first box plot (Figure 16a) shows that FT-TRENCHANT not only has the highest average of the macro metrics but is also the most stable, followed by TRENCHANT and LINE. Another standout is that box plots for specific classes (Figures 16b, 16c, 16d, 16e) show that different methods get better results on different classes. Most methods have a good performance on the *up* class, but FT-TRENCHANT is among the best, while TRENCHANT is among the methods that managed *big_down* class performance on some runs. Overall, FT-TRENCHANT has good performance throughout the macro and class-specific metrics and manages to edge out TRENCHANT and the baselines in most metrics.

We also evaluated average execution times for all algorithms on all executions for all scenarios. In Figure 17 we present the average seconds for each algorithm. It is important to denote that: (i) GCN is a semi-supervised graph neural network that is parallel on GPU; (ii) LINE is a neural network embedding method that is parallel on GPU; (iii) Struc2Vec is an extension of DeepWalk parallel on CPU; (iv) DeepWalk, Node2Vec, Metapath2Vec, TRENCHANT and, FT-TRENCHANT are sequential; and (v) all methods, except GCN, execution times include training and prediction with the LSTM. With that in mind, GCN has the fastest execution times, followed by TRENCHANT and FT-TRENCHANT. These results can be explained for two reasons: (i) they are linear methods considering nodes and edges of the network; and (ii) their stable embeddings allowed the LSTM loss to be stable sooner, resulting in a shorter training, specially FT-TRENCHANT, which has fine-tuned initial embeddings.

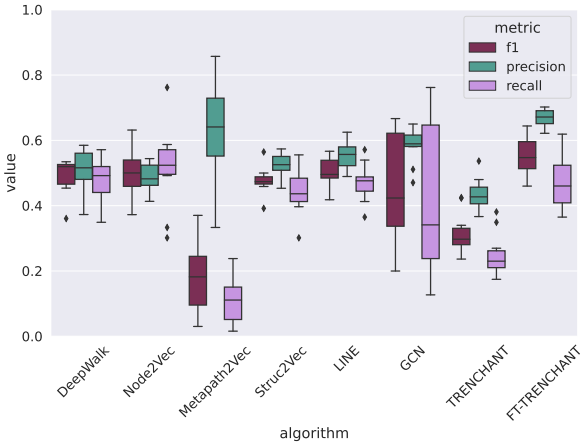
Overall, TRENCHANT and FT-TRENCHANT are methods competitive with the state-of-art network embedding methods. Also, an initial embedding ensures that all nodes will be in the same vector space. This single vector space allows new nodes added to the network to



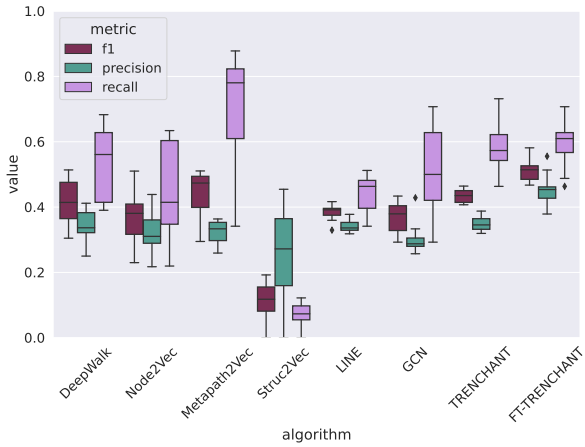
(a) macro $F1$, precision and recall.



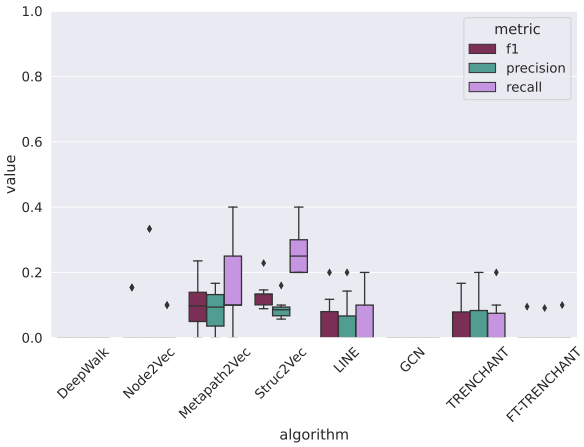
(b) *big_down* $F1$, precision and recall.



(c) *down* $F1$, precision and recall.



(d) *up* $F1$, precision and recall.



(e) *big_up* $F1$, precision and recall.

Figure 16 – Box plots for scenario #3 with the metrics: macro and class-specific $F1$, precision, and recall. Source: (CARMO; FILHO; MARCACINI, 2022).

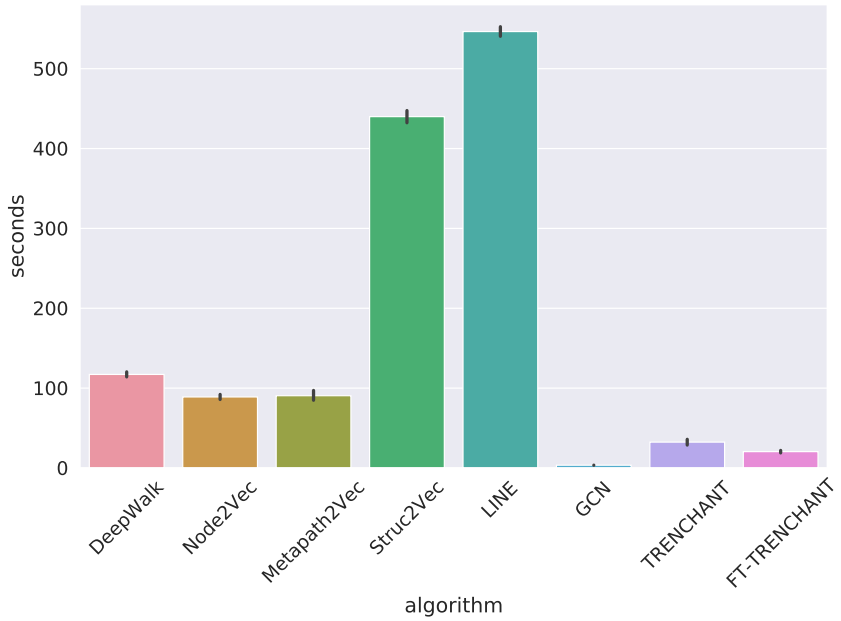


Figure 17 – Average execution times for all algorithms on all executions for each scenario in seconds. Source: (CARMO; FILHO; MARCACINI, 2022).

receive a final embedding with a few iterations on top of the existing embedding. It also provides a more stable performance since all executions will have the same starting point.

5.5 Concluding remarks

This paper introduces an embedding propagation method of pre-trained neural language models. It also proposes an extension that leverages the heterogeneous network architecture for fine-tuning a neural language model. Furthermore, we propose a pipeline for trend price prediction of agribusiness commodities prices and evaluate our proposed models against network embedding models. Finally, our experiment results show that using an embedding propagation technique from a BERT-based model allows network embedding without recalculating the entire network. We also show that the use of text information, combined with simple network topology, is competitive against state-of-art topology network embeddings algorithms when the text database is well-curated. The results with the fine-tuned extension also show that graph data can be incorporated into BERT embeddings.

We plan to incorporate weights on the different types of relations for future work, thereby allowing the embedding propagation to consider more topology information. We also want to investigate semi-supervised embedding propagation methods to incorporate the existing labels into the resulting embeddings. Finally, Attention mechanisms in place of an LSTM is another scenario we can evolve this work.

CHAPTER

6

NATUKE: NATURAL PRODUCT KNOWLEDGE EXTRACTION FROM ACADEMIC LITERATURE

This chapter presents a paper accepted in 17th IEEE International Conference on Semantic Computing (ICSC) in the resource track. The source code for the experiments and usability tips are available at <https://github.com/AKSW/natuke>. This paper was worked on during a three month guest researcher period in the Hochschule für Technik, Wirtschaft und Kultur (HTWK) at Leipzig, Germany. The papers authors were Paulo do Carmo, Edgard Marx, Ricardo Marcacini, Marilia Valli, João Victor Silva and Alan Pilon.

6.1 Initial remarks

Knowledge graphs (KGs) play a key role as a source of structured data for a variety of applications (HOGAN *et al.*, 2021). They can be specialized or multi-purpose, containing information from many domains. Nevertheless, building KGs is usually a very cumbersome and time-consuming task, often relying on manual efforts that can lead to errors or incompleteness (ZAVERI *et al.*, 2013). It is a main-fold task that involves several complex reading-understanding natural language processing techniques. An effort that can be significantly challenging when keeping the dataset up-to-date with the latest information (HELLMANN *et al.*, 2009). Devising automatic knowledge extraction methods is a far-reaching goal to facilitate KG curation and maintenance.

Machine learning (ML) methods have recently shown promising results in various natural language tasks. ML methods can cope with data nuances that can go unnoticed by rule-based approaches designed by humans over limited data observations. In this work, we introduce a crowd-sourced benchmark containing a corpus of over two thousand exemplars for evaluating

natural products' knowledge extraction from academic literature. We refer to natural products as chemical compounds generated by living organisms. They contribute as much as 67% to all drugs approved worldwide (NEWMAN; CRAGG, 2016). Natural product research relies mainly on text for academic communication. Building approaches to facilitate data querying, exploration and organization are therefore pivotal to speed up research. We also evaluate different state-of-the-art unsupervised embedding generation methods and show that it is possible to extract some properties with relatively good performance. In our evaluation, EPHEN and Metapath2Vec outperform other graph embedding methods in the natural product knowledge extraction task. Although we focus on natural products, the methods evaluated in this work can be easily extended to other domains. Overall our contributions are as follows:

- A large crowd-sourced benchmark for natural product knowledge extraction from academic literature containing over two thousand manual curated entries, and;
- A baseline evaluation of different state-of-the-art unsupervised embedding generation methods on the task of end-to-end natural product knowledge extraction from academic literature.

6.2 Related works

The US-BERT method was proposed for unsupervised relation extraction between two named entities (ALI; SALEEM; NGOMO, 2021). US-BERT uses S-BERT-type embeddings to encode sentences with hidden named entities. Then the method uses affinity propagation to create clusters based on certain sentences with greater cosine similarity with the query of entities. The method achieved better precision than the literature in all experiments.

Since this paper uses an ML-based graph completion, or entity linking, for information extraction, we must specify a few related works in this field (MARTINEZ-RODRIGUEZ; HOGAN; LOPEZ-AREVALO, 2020). The NED-EE method is based on conditional random fields (CRF) classifier and combines other non-ML techniques like Stanford NER in its structure (HOFFART; ALTUN; WEIKUM, 2014). ADEL (PLU; RIZZO; TRONCY, 2016) and UDFS (DERCZYNSKI; AUGENSTEIN; BONTCHEVA, 2015) also use Stanford NER, while JERL (LUO *et al.*, 2015) uses a custom CRF model that recognizes and links to existing KGs. WAT relies on OpenNLP's NER in a maximum entropy model (PICCINNO; FERRAGINA, 2014). On the other hand, J-NERD (NGUYEN; THEOBALD; WEIKUM, 2016) uses Stanford's dependency parse-tree in each sentence and then combines it into a global model used in an inference model based on Gibbs sampling.

Although these methods achieve good performance, they are limited by the vocabulary they are trained in. In order to breach this limitation, we search for unsupervised methods that allow for the dynamic insertion of new nodes or at least reduced time execution for the

information extraction process. Therefore, we propose this benchmark to evaluate four state-of-art unsupervised graph embedding models in the task of natural product knowledge extraction from academic literature.

6.3 Problem definition

Our benchmark NatUKE aims to evaluate natural product knowledge extraction from academic literature. It considers three main aspects of this task: (A) we use the natural product data set NuBBE$_{DB}$ (PILON *et al.*, 2017) to evaluate the extraction of some characteristics it considers from papers; (B) the knowledge extraction results are obtained through KG completion by obtaining similarity distances from unsupervised graph embedding models; and (C) we evaluate four different graph embedding models to compare how their structures behave throughout the data set.

6.3.1 Dataset curation

The dataset used for evaluation and training was generated from hundreds of peer-reviewed scientific articles with information on more than 2,521 possibilities of natural product extraction. The dataset was built manually by chemistry specialists that read the articles annotating four relevant properties associated with each natural product discussed in the academic publication. In this work, we focus on five NuBBE$_{DB}$ properties for training and prediction: (I) compound name (rdfs:label), (II) bioactivity (nubbe:biologicalActivity), (III) species from where natural products were extracted (nubbe:collectionSpecie), (IV) collection site of these species (nubbe:collectionSite), and (V) isolation type (nubbe:collectionType). Table **??** presents an overview of the number of unique properties.

Table 14 – Overview of the number of distinct values per property.

#Compounds	#Bioactivities	#Species	#Sites	#Isolations
446	34	116	52	6

6.3.2 Experimental setup & evaluation criteria

The problem of knowledge extraction from unstructured data sources is that, even with academic papers as input data, authors may use different words to describe the same subjects, methods, or techniques. Using rule-based information extraction algorithms is challenging even though it tends to output more stable and trustworthy results.

In this work, we propose a benchmark and evaluate different ML graph embeddings for the task of unsupervised knowledge extraction. Graph embeddings allow the extraction of information already present in the graph by previous extractions and encode them in characteristic

vectors. In order to use graph embedding methods, we model a KG with the paper's DOI as a central node (Figure 18). It connects to the previously extracted characteristics, such as related molecule data properties and topics extracted from BERTopic (GROOTENDORST, 2020). BERTopic is a topic modeling technique based on the BERT (Bidirectional Encoder Representations from Transformers) (DEVLIN *et al.*, 2018) method and a class-based TF-IDF to create dense clusters while keeping important words from objects in these clusters to allow for easily readable topics.

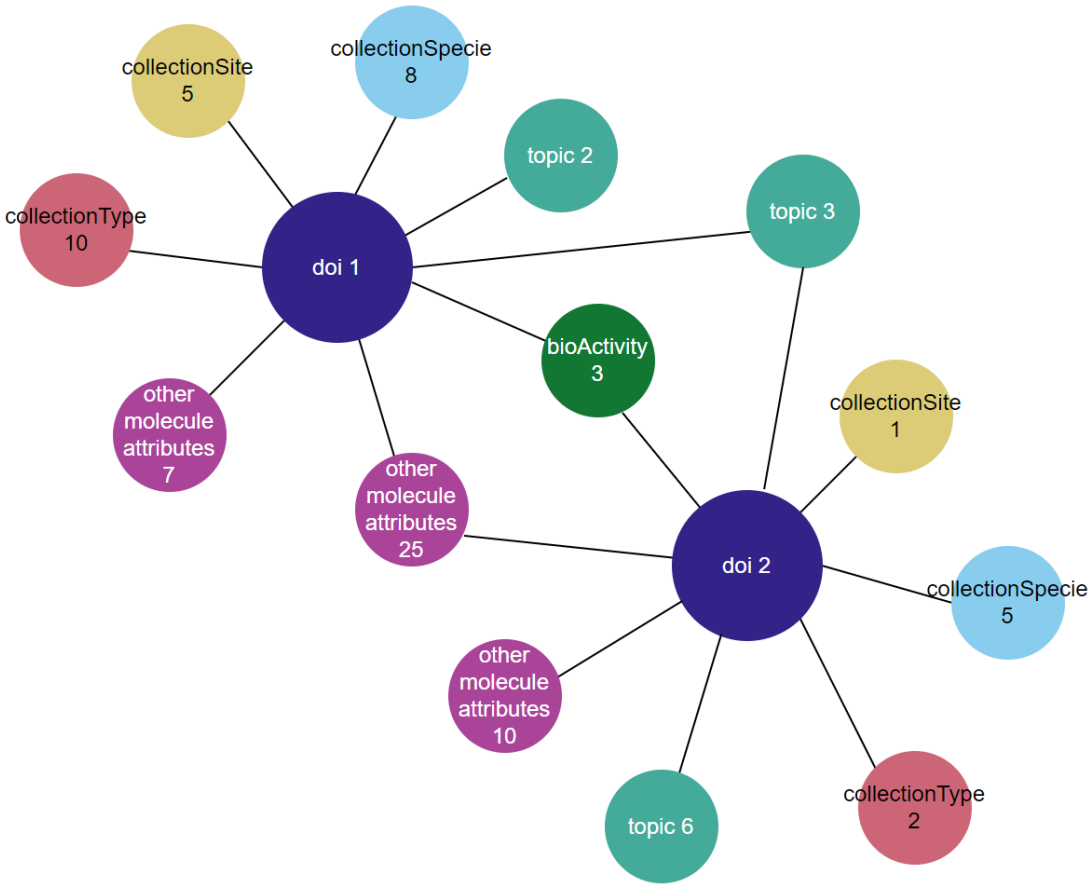


Figure 18 – Example of the proposed structure for the KG.

The BERTopic model requires a certain number of tokens that do not allow the insertion of entire papers. Therefore, we split the papers into sentences and fed them to the BERTopic model. So, we fit all the sentences to a multilingual pre-trained BERTopic model and extract non-duplicate topics. We filter the paper's topics by allowing connection only when less than 80% of the papers were previously connected. That allows us to eliminate general topics that do not discriminate papers enough automatically. With them, we can hide all links to manually extracted characteristics whenever a paper is selected as testing data while maintaining them connected to other nodes by their topics.

In order to simulate a scenario where new training data is constantly added to the model, we modeled a dynamic evaluation of four stages with different amounts of training data in each stage. The first train/test split consists of a resp. 20/80% ratio, and for other stages, the train split

is increased by 20% until it reaches an 80/20% ratio (Figure 19). In our benchmark, we evaluate the accuracy of each approach in predicting the resource from different chemical compound properties using `hits@k`. The metric `hits@k` measures the average of how many predictions achieve top `k` rankings (DOCS, 2019). Together with MRR (Mean Reciprocal Rank), `hits@k` is a ranking metric for when there is only one correct document. On the other hand, mAP (mean Average Precision) and nDCG (normalized Discounted Cumulative Gain) (KISHIDA, 2005) are designed for ranking when a list of relevant documents is available. We chose `hits@k` because it allows us to evaluate each characteristic extraction with reasonable expectations by customizing the *k* value.

To coupe with the principles of Findable, Accessible, Interoperable, and Reusable (FAIR)[1] and to facilitate reproducibility, all experiments and data are publicly available at <http://github.com/AKSW/natuke> under Apache License 2.0.[2]

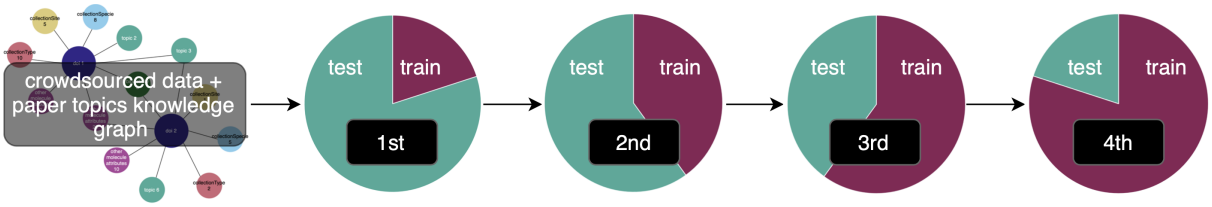


Figure 19 – Dynamic evaluation stages for evaluation. Source: (CARMO *et al.*, 2023).

6.3.3 Models & Frameworks

We compare four different unsupervised graph embedding methods for our knowledge extraction task: (1) DeepWalk (PEROZZI; AL-RFOU; SKIENA, 2014) is an unsupervised graph embedding method that uses random walks to sample a training dataset for a skip-gram architecture; (2) Node2Vec (GROVER; LESKOVEC, 2016) extends the DeepWalk method to allow more control on the random walks; (3) Metapath2Vec (DONG; CHAWLA; SWAMI, 2017) is another extension from DeepWalk that transforms the random walks into meta-path-based walks; and (4) Embedding Propagation on Heterogeneous Networks (EPHEN) (CARMO; MARCACINI, 2021) is an embedding propagation method that uses a regularization function to distribute an initial embedding on a KG. Therefore it considers both text and structured data in an unsupervised scenario. EPHEN relies on a startup embedding, so we used the sentences generated for the BERTopic and fed them individually to a Sentence-BERT (REIMERS; GUREVYCH, 2019) multilingual model. The final paper startup embedding is the sum of all the sentence embeddings, which shifts the vector for a paper representation. We also tried to evaluate GraphSAGE (HAMILTON; YING; LESKOVEC, 2017). However, our dataset does not achieve a complete KG, and GraphSAGE's neural networks can not propagate the features and generate new embeddings without a complete graph.

[1] <https://www.go-fair.org/fair-principles/>
[2] <https://www.apache.org/licenses/LICENSE-2.0>

We chose these methods because they generate embeddings for every node without needing a complete KG, pre-determined weights, or ontology. That is important to us since these characteristics are necessary for applying the model in the real world since papers will be connected only by automatically extracted characteristics. We are also interested in dynamic models that can generate these embeddings within a reasonable amount of time and computational resources.

6.4 Experimental results

Table 15 shows the results from experiments extracting five different natural product properties from biochemical academic papers. We use NuBBE$_{KG}$ ontology and dataset[3] for property prediction. We use different values of k proportionally to the property-value prediction challenge. For instance, it is harder to predict the correct natural product name than it is to predict the isolation type. As shown in Table 14, there are considerably fewer unique possible characteristics for isolation type than compound name. Therefore predicting the correct compound name is significantly more challenging. The final k values in this table are from 1 to 50 considering values multiples of 5 and two thresholds: (1) a score equal or higher than 0.50 is achieved; and (2) a score equal or higher than 0.20 is achieved. Considering these conditions, compound name (C) and isolation type (T) achieve the same final k value in both thresholds. While bioactivity (B), collection species (S), and collection sites (L) have different k values for each threshold.

Table 15 – Results table for extracting: compound name (C), bioactivity (B), specie (S), collection site (L), and isolation type (T). The results consider different final k values corresponding to two different rules. The best results for each extraction are bold.

Property	k	**DeepWalk**				**Node2Vec**				**Metapath2Vec**				**EPHEN**			
		1st	**2nd**	**3rd**	**4th**	**1st**	**2nd**	**3rd**	**4th**	**1st**	**2nd**	**3rd**	**4th**	**1st**	**2nd**	**3rd**	**4th**
C	50	0.08	0.00	0.00	0.00	0.08	0.00	0.00	0.00	**0.10**	**0.08**	**0.09**	**0.20**	0.09	0.02	0.03	0.04
B	5	0.41	0.12	0.10	0.07	0.41	0.07	0.03	0.03	0.27	0.17	0.13	0.12	**0.55**	**0.57**	**0.60**	**0.64**
	1	0.10	0.01	0.01	0.01	0.09	0.02	0.01	0.01	0.06	0.04	0.03	0.10	**0.17**	**0.19**	**0.24**	**0.25**
S	50	0.37	0.24	0.27	0.25	0.36	0.22	0.25	0.24	**0.40**	**0.41**	**0.42**	**0.44**	0.36	0.24	0.29	0.30
	20	0.10	0.12	0.12	0.11	0.10	0.13	0.11	0.11	**0.15**	0.11	0.15	0.19	0.10	**0.15**	**0.19**	**0.22**
L	20	0.56	0.41	0.38	0.29	**0.57**	0.36	0.28	0.23	0.40	0.42	0.42	0.40	0.53	**0.52**	**0.55**	**0.55**
	5	0.15	0.09	0.06	0.06	0.13	0.08	0.06	0.05	0.12	0.13	0.11	0.13	**0.26**	**0.29**	**0.30**	**0.27**
T	1	0.25	0.14	0.14	0.09	0.10	0.07	0.05	0.01	0.28	0.22	0.19	0.19	**0.71**	**0.66**	**0.75**	**0.75**

Overall, EPHEN achieves the best bioactivity and isolation type extraction performance, increasing the accuracy through the evaluation stages. For example, in the bioactivity extraction, EPHEN achieves 0.55 `hits@5` in the first evaluation stage and progressively better results until 0.64 in the fourth evaluation stage. In contrast, DeepWalk has the second-best results on the first evaluation with 0.41 and drops performance until the second-worst results with 0.07 in the fourth evaluation stage.

[3] <http://nubbe.aksw.org>

Metapath2Vec achieves the best accuracy in extracting compound names and species when $k = 50$. However, EPHEN performs best from the second evaluation stage onward for collection species when $k = 20$. That shows us that EPHEN's embeddings allow sorting the correct document in the first rankings more often than Metapath2Vec's embeddings. Meanwhile, Metapath2Vec catches up to the performance when considering longer k intervals. That shows us that Metapath2Vec's heterogeneous type data extraction helps to map global relationships that are not possible in the other evaluated methods.

Table 16 shows the execution times for each extraction scenario. Metapath2Vec benefits from its parallel execution in the first two steps, which contain more examples in the test split. Meanwhile, EPHEN's dynamic structure allows embedding generation in the same vector space, meaning it does not have to re-calculate embeddings from scratch. That reduces the final execution time significantly after the first step, which grants EPHEN the best execution times after the third step. We argue that it also allows EPHEN to improve performance after the first evaluation stage better than the other evaluated methods, as shown in Table 15.

Table 16 – Execution times table for extracting: compound name (C), bioactivity (B), specie (S), collection site (L), and isolation type (T). All time executions were measured in seconds. The lowest time executions for each extraction are bold.

Property	DeepWalk				Node2Vec				Metapath2Vec				EPHEN			
	1st	2nd	3rd	4th	1st	2nd	3rd	4th	1st	2nd	3rd	4th	1st	2nd	3rd	4th
C	62.5	50.1	43.5	40.5	59.7	43.7	33.7	28.4	**51.5**	**32.3**	21.5	14.1	58.0	33.6	**21.0**	**11.7**
B	33.8	35.5	35.3	36.3	30.1	27.3	25.8	24.7	**24.7**	**18.1**	14.4	11.6	27.6	18.4	**13.5**	**9.32**
S	29.5	32.7	34.8	36.8	30.0	25.5	25.2	24.7	**19.5**	**16.2**	14.2	11.9	23.5	16.7	**13.3**	**9.71**
L	30.1	33.0	35.4	37.1	27.5	28.4	25.4	25.3	**20.0**	**16.4**	13.8	12.0	23.9	17.0	**13.1**	**9.59**
T	29.4	32.8	36.7	39.5	26.1	26.2	25.3	25.5	**19.0**	**16.4**	13.9	12.4	23.0	17.0	**12.8**	**9.69**

6.5 Concluding remarks

Our evaluation shows that specific unsupervised graph embedding methods can be used for certain characteristics to extract natural product knowledge from academic literature through KG completion. Particularly the properties with fewer candidates (i.e., bioactivity and isolation type). In most cases, using context-aware data leads to improvements in extraction quality. EPHEN has achieved the best results, while Metapath2Vec showed good performance in more challenging scenarios (i.e., chemical compound and collection site prediction). Finally, DeepWalk and Node2Vec's random walks perform better with fewer training corpora. All methods evaluated achieve good execution times in this dataset when considering execution times. Meanwhile, EPHEN's capability to dynamically update embeddings shows an excellent trend to better handle more extensive datasets in a real-world scenario, where the automatized extraction will help increase the datasets' size. In future work, we plan to fine-tune those models using resource similarity data and develop an automatic natural product knowledge extraction framework with

the human-in-the-loop. We also plan to increase the robustness of the KGs by adding data related to the papers like citation networks, authors, publishers, and institutions related.

CHAPTER

7

CONCLUSION

This chapter presents the conclusion of this dissertation. First, the contributions and scientific innovations are presented, referring to the method and the answers to the proposed challenges and goals. Second, the publications and the research collaboration resulting from this masters are presented. Finally, the limitations of the proposed method and future work are discussed.

7.1 Contributions and scientific innovations

This master's dissertation presents an unsupervised embedding propagation method for heterogeneous information networks that maintain a fixed vector space regarding the start-up embedding, which makes the dynamic insertion of new nodes possible. The embedding propagation method was evaluated in heterogeneous information networks modeled after text data, using the SBERT (REIMERS; GUREVYCH, 2019) embedding model as the starting vector in different applications such as: (i) an event analysis task that predict related events, actors, and locations with graph completion; (ii) a trend price prediction model for the soybean and corn commodities in the Brazilian market; and (iii) an information extraction of chemical compound characteristics from academic papers in partnership with a German Institution. The innovations presented in this master's dissertation are highlighted below:

- **Embedding propagation over heterogeneous information networks**: we proposed and developed an unsupervised embedding propagation method. The method is called Embedding propagation over heterogeneous information networks (EPHEN), and it propagates an initial embedding through a regularization function. It can propagate in both homogeneous and heterogeneous information networks. Since it is an unsupervised method, it generates generic embeddings that can be used throughout different tasks without any modifications.

As a result, EPHEN outperforms several other unsupervised network embedding methods from the literature.

- **Embedding propagation over heterogeneous event networks**: is the use case paper highlighted in Chapter 4 where EPHEN was applied in event analysis. That paper presents EPHEN compared to other network embedding methods in three tasks of link prediction on heterogeneous event networks modeled after events Project GDELT created after news published in international portals. Each central event node has an SBERT embedding vector associated and is connected to other attributes from this dataset: published date, location, actors, and themes. EPHEN shows better performance than any other network embedding method used as baselines. It also achieves second-best execution times, losing only to a GPU parallel method. This paper also shows how EPHEN can dynamically insert new nodes into the vector space.

- **Commodities trend prediction on heterogeneous information networks**: Chapter 5 shows an extended paper on the trend price prediction of soybean and corn commodities use case. This paper uses commodities-related news and the 5W1H (HAMBORG *et al.*, 2018) characteristics as well as weekly and monthly swings of price to model heterogeneous information networks for a trend link prediction task. This paper also shows a LSTM neural network as the classifier for a multi-class classification of the price swing (*big_down*, *down*, *up*, *big_up*). Since this chapter presents the extended version of the paper, it also presents an unsupervised fine-tuning pipeline for the SBERT that is also compared against previous baselines and the pre-trained version of the SBERT model as the start-up embedding. This paper shows exciting results such as better performances from fine-tuned and pre-trained methods in different scenarios. The fine-tuning helped to achieve better performance from other network embedding methods when the propagation of pre-trained language model embedding performed worse than topological network embedding methods.

- **NatUKE: A benchmark for natural product knowledge extraction from academic literature**: this paper shows a benchmark of different unsupervised network embedding on the task of information extraction of chemical compound characteristics from academic papers. This paper was developed in a collaboration work with the Hochschule für Technik, Wirtschaft und Kultur (HTWK) at Leipzig, Germany. The paper aimed to evaluate if unsupervised network embedding methods and graph completion techniques could be used for information extraction (IE). The IE process is then used to automatize the update process of a Resource Description Framework (RDF) knowledge graph of chemical compounds. EPHEN showed competitive performance compared to other network embedding methods in most scenarios. Allied with EPHEN's ability to dynamically insert new nodes to the same vector space allows a human-in-loop system implementation.

7.2 Publications

Publications in conferences and one journal helped to disseminate the obtained throughout the development of this master's. A list of publications separated by type is presented below:

- **Conferences**

  1. CARMO, P. do; FILHO, I. R.; MARCACINI, R. Commodities trend link prediction on heterogeneous information networks. In: SBC. **Anais do IX Symposium on Knowledge Discovery, Mining, and Learning**, 2021. p. 81–88.
  2. CARMO, P. do; MARCACINI, R. Embedding propagation over heterogeneous event networks for link prediction. In: **2021 IEEE International Conference on Big Data (Big Data)**, 2021. p. 4812–4821.

- **Accepted in Journals**

  1. CARMO, P. do; FILHO, I. R.; MARCACINI, R. TRENCHANT: TRENd prediCtion on Heterogeneous informAtion NeTworks. In: **Journal of Information and Data Management**, 2022. (prelo)

- **Accepted in Conferences**

  1. CARMO, P. do; MARX, E.; MARCACINI, R.; VALLI, M.; SILVA, J. V.; PILON, A. NatUKE: A Benchmark for Natural Product Knowledge Extraction from Academic Literature. In: **17th IEEE International Conference on Semantic Computing**, 2023. (prelo)

It is important to reinforce that all papers have the source code, datasets, and some usability guidelines are publicly available at the following addresses, respectively: <https://github.com/PauloRVdC/tphin-experiments>; <https://github.com/PauloRVdC/ephen-experiments>; <https://github.com/PauloRVdC/TRENCHANT>; and <https://github.com/AKSW/natuke>.

7.3 Limitations and future work

The proposed embedding propagation method fulfilled the research goals of propagating the initial embedding from some textual nodes to the remaining nodes in a heterogeneous information network and allowing dynamic insertion of new nodes in the embedding propagation process.

This master's dissertation explored one regularization function for propagating the embeddings that adapted itself throughout different domains and tasks. Even though the proposed

regularization function achieved the research goals of propagating text embeddings while allowing dynamic generation of new embeddings to nodes connected afterwards, no other function was proposed and evaluated in comparison.

Another limitation of the experiments was that the results of a general knowledge pre-trained start-up embedding were not compared to domain-specific fine-tunings. Even though that was the case, the master's dissertation also proposes and evaluates an unsupervised fine-tuning pipeline for the used SBERT structure compared to the pre-trained model.

Statistical analysis for the proposed method against baselines were also not executed because few datasets were used within each use case. Therefore, statistical methods would not provide enough information to discriminate the methods based on their performance. However, the proposed method was evaluated against baselines in three different use cases and it showed promising performance within these limited analysis.

For future work, the aim is to solve the current limitations. Another possibility is to include linguistics and semantic features within the network and fine-tuning stages to try and boost the regularization performance. Moreover, they must also consider other methods of graph neural networks, besides GCN, for baseline comparison. Graph neural networks can also be used to extend the research, for example: the current embedding propagation method can only use one type of start-up embeddings at a time, and if a HIN is modeled after text and images, the proposed method can only propagate text or image features to the remaining nodes. Graph neural networks allow different types of features to be joined together in an end-to-end pipeline that uses the regularization function to generate features of all types to all nodes in a network.

BIBLIOGRAPHY

ADANACIOGLU, H.; YERCAN, M. *et al.* An analysis of tomato prices at wholesale level in turkey: an application of sarima model. **Custos e Agronegócio Online**, v. 8, n. 4, p. 52–75, 2012. Citation on page 72.

AGGARWAL, C. C. **Machine learning for text**. "": Springer, 2018. Citations on pages 21, 29, 34, and 36.

AGGARWAL, C. C.; ZHAI, C. **Mining text data**. "": Springer Science & Business Media, 2012. Citations on pages 21, 27, and 29.

ALI, M.; SALEEM, M.; NGOMO, A.-C. N. Unsupervised relation extraction using sentence encoding. In: SPRINGER. **European Semantic Web Conference**. [S.l.], 2021. p. 136–140. Citation on page 82.

ALLAN, J. **Topic detection and tracking: event-based information organization**. "": Springer Science & Business Media, 2012. Citations on pages 59 and 69.

ALPAYDIN, E. **Introduction to machine learning**. "": MIT press, 2020. Citation on page 34.

BERTON, L.; LOPES, A. D. A. Graph construction based on labeled instances for semi-supervised learning. In: IEEE. **2014 22nd International Conference on Pattern Recognition**. "", 2014. p. 2477–2482. Citation on page 48.

BOURLARD, H.; KAMP, Y. Auto-association by multilayer perceptrons and singular value decomposition. **Biological cybernetics**, Springer, v. 59, n. 4, p. 291–294, 1988. Citation on page 34.

BROWNLEE, J. A gentle introduction to the rectified linear unit (relu). **Machine learning mastery**, v. 6, 2019. Citation on page 35.

CARMO, P. do; FILHO, I. R.; MARCACINI, R. Commodities trend link prediction on heterogeneous information networks. In: SBC. **Anais do IX Symposium on Knowledge Discovery, Mining and Learning**. [S.l.], 2021. p. 81–88. Citations on pages 16, 24, 69, 71, and 73.

____. Trenchant: Trend prediction on heterogeneous information networks (prelo). **Journal of Information and Data Management**, 2022. Citations on pages 16, 24, 73, 78, and 79.

CARMO, P. do; MARCACINI, R. Embedding propagation over heterogeneous event networks for link prediction. In: **2021 IEEE International Conference on Big Data (Big Data)**. [S.l.: s.n.], 2021. p. 4812–4821. Citations on pages 16, 24, 59, 62, 66, and 85.

CARMO, P. do; MARX, E.; MARCACINI, R.; VALLI, M.; SILVA, J.; PILON, A. Natuke: Natural product knowledge extraction from academic literature (prelo). **17th IEEE International Conference on Semantic Computing**, 2023. Citations on pages 16, 24, and 85.

CHANG, S.; HAN, W.; TANG, J.; QI, G.-J.; AGGARWAL, C. C.; HUANG, T. S. Heterogeneous network embedding via deep architectures. In: **Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. "": "", 2015. p. 119–128. Citations on pages 22, 60, and 70.

CHEN, H.; YIN, H.; WANG, W.; WANG, H.; NGUYEN, Q. V. H.; LI, X. Pme: projected metric embedding on heterogeneous networks for link prediction. In: **Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. "": "", 2018. p. 1177–1186. Citation on page 49.

CHEN, H.-H.; CHEN, M.; CHIU, C.-C. The integration of artificial neural networks and text mining to forecast gold futures prices. **Communications in Statistics - Simulation and Computation**, Taylor Francis, v. 45, n. 4, p. 1213–1225, 2016. Citation on page 72.

CHEN, X.; LI, Q. Event modeling and mining: a long journey toward explainable events. **The VLDB Journal**, Springer, v. 29, n. 1, p. 459–482, 2020. Citations on pages 22, 59, 69, and 70.

CORDEIRO, M.; GAMA, J. Online social networks event detection: a survey. In: **Solving Large Scale Learning Tasks. Challenges and Algorithms**. "": Springer, 2016. p. 1–41. Citations on pages 59 and 69.

CRASWELL, N. Mean reciprocal rank. In: ____. **Encyclopedia of Database Systems**. Boston, MA: Springer US, 2009. p. 1703–1703. ISBN 978-0-387-39940-9. Available: <https://doi.org/10.1007/978-0-387-39940-9_488>. Citation on page 37.

CUI, P.; WANG, X.; PEI, J.; ZHU, W. A survey on network embedding. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 31, n. 5, p. 833–852, 2018. Citations on pages 22, 43, 60, and 70.

DAREKAR, A.; REDDY, A. Predicting market price of soybean in major india studies through arima model. **Journal of Food Legumes**, v. 30, n. 2, p. 73–76, 2017. Citation on page 72.

DEFFERRARD, M.; BRESSON, X.; VANDERGHEYNST, P. Convolutional neural networks on graphs with fast localized spectral filtering. **Advances in neural information processing systems**, v. 29, p. 3844–3852, 2016. Citation on page 46.

DENG, S.; RANGWALA, H.; NING, Y. Learning dynamic context graphs for predicting social events. In: **Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. "": "", 2019. p. 1007–1016. Citations on pages 48, 59, and 71.

____. Dynamic knowledge graph based multi-event forecasting. In: **Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. "": "", 2020. p. 1585–1595. Citations on pages 49 and 71.

DERCZYNSKI, L.; AUGENSTEIN, I.; BONTCHEVA, K. Usfd: Twitter ner with drift compensation and linked data. **arXiv preprint arXiv:1511.03088**, 2015. Citation on page 82.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018. Citations on pages 15, 21, 22, 28, 29, 31, 33, 34, 60, 70, and 84.

DOCS, A. **Hits at n score**. 2019. Citations on pages 37 and 85.

DONG, Y.; CHAWLA, N. V.; SWAMI, A. metapath2vec: Scalable representation learning for heterogeneous networks. In: **Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining**. "": "", 2017. p. 135–144. Citations on pages 44 and 85.

DURAN, A. G.; NIEPERT, M. Learning graph representations with embedding propagation. In: **Advances in neural information processing systems**. "": "", 2017. p. 5119–5130. Citations on pages 22 and 47.

ENGELEN, J. E. V.; HOOS, H. H. A survey on semi-supervised learning. **Machine Learning**, Springer, v. 109, n. 2, p. 373–440, 2020. Citation on page 39.

FILHO, I. J. dos R.; CORREA, G. B.; FREIRE, G. M.; REZENDE, S. O. Forecasting future corn and soybean prices: an analysis of the use of textual information to enrich time-series. In: SBC. **Anais do VIII Symposium on Knowledge Discovery, Mining and Learning**. "", 2020. p. 113–120. Citation on page 70.

FLORENCE, R.; NOGUEIRA, B.; MARCACINI, R. Constrained hierarchical clustering for news events. In: **Proceedings of the 21st International Database Engineering & Applications Symposium**. "": "", 2017. p. 49–56. Citation on page 59.

GHAG, K. V.; SHAH, K. Comparative analysis of effect of stopwords removal on sentiment classification. In: IEEE. **2015 international conference on computer, communication and control (IC4)**. "", 2015. p. 1–6. Citation on page 29.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. 6.2. 2.3 softmax units for multinoulli output distributions. **Deep learning**, MIT press, n. 1, p. 180, 2016. Citations on pages 35 and 74.

GROOTENDORST, M. **BERTopic: Leveraging BERT and c-TF-IDF to create easily interpretable topics.** [S.l.]: Zenodo, 2020. Citation on page 84.

GROVER, A.; LESKOVEC, J. node2vec: Scalable feature learning for networks. In: **Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining**. "": "", 2016. p. 855–864. Citations on pages 44 and 85.

GUI, H.; LIU, J.; TAO, F.; JIANG, M.; NORICK, B.; KAPLAN, L.; HAN, J. Embedding learning with events in heterogeneous information networks. **IEEE transactions on knowledge and data engineering**, IEEE, v. 29, n. 11, p. 2428–2441, 2017. Citation on page 47.

HAMBORG, F.; LACHNIT, S.; SCHUBOTZ, M.; HEPP, T.; GIPP, B. Giveme5w: main event retrieval from news articles by extraction of the five journalistic w questions. In: SPRINGER. **International Conference on Information**. "", 2018. p. 356–366. Citations on pages 59, 70, 72, and 90.

HAMILTON, W.; YING, Z.; LESKOVEC, J. Inductive representation learning on large graphs. **Advances in neural information processing systems**, v. 30, 2017. Citations on pages 22, 60, and 85.

HAMMOND, D. K.; VANDERGHEYNST, P.; GRIBONVAL, R. Wavelets on graphs via spectral graph theory. **Applied and Computational Harmonic Analysis**, Elsevier, v. 30, n. 2, p. 129–150, 2011. Citation on page 46.

HAN, J.; MORAGA, C. The influence of the sigmoid function parameters on the speed of back-propagation learning. In: SPRINGER. **International workshop on artificial neural networks**. "", 1995. p. 195–201. Citation on page 35.

HELLMANN, S.; STADLER, C.; LEHMANN, J.; AUER, S. DBpedia live extraction. In: SPRINGER. **OTM Confederated International Conferences**. [S.l.], 2009. p. 1209–1223. Citation on page 81.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citation on page 35.

HOFFART, J.; ALTUN, Y.; WEIKUM, G. Discovering emerging entities with ambiguous names. In: **Proceedings of the 23rd international conference on World wide web**. [S.l.: s.n.], 2014. p. 385–396. Citation on page 82.

HOGAN, A.; BLOMQVIST, E.; COCHEZ, M.; D'AMATO, C.; MELO, G. d.; GUTIERREZ, C.; KIRRANE, S.; GAYO, J. E. L.; NAVIGLI, R.; NEUMAIER, S. *et al.* Knowledge graphs. **Synthesis Lectures on Data, Semantics, and Knowledge**, Morgan & Claypool Publishers, v. 12, n. 2, p. 1–257, 2021. Citation on page 81.

HU, Z.; DONG, Y.; WANG, K.; SUN, Y. Heterogeneous graph transformer. In: **Proceedings of The Web Conference 2020**. [S.l.: s.n.], 2020. p. 2704–2710. Citation on page 60.

HUANG, Z.; MAMOULIS, N. Heterogeneous information network embedding for meta path based proximity. **arXiv preprint arXiv:1701.05291**, 2017. Citations on pages 22, 60, and 70.

JACCARD, P. Étude comparative de la distribution florale dans une portion des alpes et des jura. **Bull Soc Vaudoise Sci Nat**, v. 37, p. 547–579, 1901. Citation on page 48.

JI, M.; SUN, Y.; DANILEVSKY, M.; HAN, J.; GAO, J. Graph regularized transductive classification on heterogeneous information networks. In: SPRINGER. **Joint European Conference on Machine Learning and Knowledge Discovery in Databases**. "", 2010. p. 570–586. Citations on pages 41 and 54.

KIPF, T. N.; WELLING, M. Semi-supervised classification with graph convolutional networks. In: **International Conference on Learning Representations (ICLR)**. "": "", 2017. Citation on page 46.

KISHIDA, K. **Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments**. "": National Institute of Informatics Tokyo, Japan, 2005. Citation on page 85.

KUMAR, A.; SINGH, S. S.; SINGH, K.; BISWAS, B. Link prediction techniques, applications, and performance: A survey. **Physica A: Statistical Mechanics and its Applications**, Elsevier, v. 553, p. 124289, 2020. Citation on page 60.

LI, J.-c.; ZHAO, D.-l.; GE, B.-F.; YANG, K.-W.; CHEN, Y.-W. A link prediction method for heterogeneous networks based on bp neural network. **Physica A: Statistical Mechanics and its Applications**, Elsevier, v. 495, p. 1–17, 2018. Citation on page 34.

LIBEN-NOWELL, D.; KLEINBERG, J. The link-prediction problem for social networks. **Journal of the American society for information science and technology**, Wiley Online Library, v. 58, n. 7, p. 1019–1031, 2007. Citation on page 60.

LUO, G.; HUANG, X.; LIN, C.-Y.; NIE, Z. Joint entity recognition and disambiguation. In: **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing**. [S.l.: s.n.], 2015. p. 879–888. Citation on page 82.

MARCACINI, R. M.; ROSSI, R. G.; NOGUEIRA, B. M.; MARTINS, L. V.; CHERMAN, E. A.; REZENDE, S. O. Websensors analytics: Learning to sense the real world using web news events. In: **Simp. Brasileiro de Sistemas Multimídia e Web**. "": "", 2017. p. 169–173. Citations on pages 59 and 69.

MARTINEZ-RODRIGUEZ, J. L.; HOGAN, A.; LOPEZ-AREVALO, I. Information extraction meets the semantic web: a survey. **Semantic Web**, IOS Press, v. 11, n. 2, p. 255–335, 2020. Citation on page 82.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. S.; DEAN, J. Distributed representations of words and phrases and their compositionality. In: **Advances in neural information processing systems**. "": "", 2013. p. 3111–3119. Citations on pages 15, 21, 22, 28, 29, 30, 31, and 34.

MORIN, F.; BENGIO, Y. Hierarchical probabilistic neural network language model. In: PMLR. **International workshop on artificial intelligence and statistics**. "", 2005. p. 246–252. Citation on page 30.

NEWMAN, D. J.; CRAGG, G. M. Natural products as sources of new drugs from 1981 to 2014. **Journal of natural products**, ACS Publications, v. 79, n. 3, p. 629–661, 2016. Citation on page 82.

NEWMAN, M. **Networks: An Introduction**. Oxford University Press, 2010. ISBN 9780199206650. Available: <https://doi.org/10.1093/acprof:oso/9780199206650.001.0001>. Citation on page 38.

NGUYEN, D. B.; THEOBALD, M.; WEIKUM, G. J-nerd: joint named entity recognition and disambiguation with rich linguistic features. **Transactions of the Association for Computational Linguistics**, MIT Press, v. 4, p. 215–229, 2016. Citation on page 82.

NING, Y.; MUTHIAH, S.; RANGWALA, H.; RAMAKRISHNAN, N. Modeling precursors for event forecasting via nested multi-instance learning. In: **Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining**. "": "", 2016. p. 1095–1104. Citation on page 48.

NING, Y.; ZHAO, L.; CHEN, F.; LU, C.-T.; RANGWALA, H. Spatio-temporal event forecasting and precursor identification. In: **Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. "": "", 2019. p. 3237–3238. Citations on pages 59 and 70.

OZCAN, A.; OGUDUCU, S. G. Multivariate time series link prediction for evolving heterogeneous network. **International Journal of Information Technology & Decision Making**, World Scientific, v. 18, n. 01, p. 241–286, 2019. Citation on page 49.

PEROZZI, B.; AL-RFOU, R.; SKIENA, S. Deepwalk: Online learning of social representations. In: **Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining**. "": "", 2014. p. 701–710. Citations on pages 43 and 85.

PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. Deep contextualized word representations. **arXiv preprint arXiv:1802.05365**, 2018. Citation on page 31.

PICCINNO, F.; FERRAGINA, P. From tagme to wat: a new entity annotator. In: **Proceedings of the first international workshop on Entity recognition & disambiguation**. [S.l.: s.n.], 2014. p. 55–62. Citation on page 82.

PILON, A. C.; VALLI, M.; DAMETTO, A. C.; PINTO, M. E. F.; FREIRE, R. T.; CASTRO-GAMBOA, I.; ANDRICOPULO, A. D.; BOLZANI, V. S. Nubbedb: an updated database to uncover chemical and biological information from brazilian biodiversity. **Scientific Reports**, Nature Publishing Group, v. 7, n. 1, p. 1–12, 2017. Citation on page 83.

PLU, J.; RIZZO, G.; TRONCY, R. Enhancing entity linking by combining ner models. In: SPRINGER. **Semantic Web Evaluation Challenge**. [S.l.], 2016. p. 17–32. Citation on page 82.

POWERS, D. Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation. **Mach. Learn. Technol.**, v. 2, 01 2008. Citations on pages 36 and 37.

RADINSKY, K.; HORVITZ, E. Mining the web to predict future events. In: **Proceedings of the sixth ACM international conference on Web search and data mining**. "": "", 2013. p. 255–264. Citations on pages 59 and 70.

REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In: **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)**. [S.l.: s.n.], 2019. p. 3982–3992. Citations on pages 53, 55, 85, and 89.

____. Making monolingual sentence embeddings multilingual using knowledge distillation. **arXiv preprint arXiv:2004.09813**, 2020. Citation on page 53.

REZENDE, S. O. **Sistemas inteligentes: fundamentos e aplicações**. "": Editora Manole Ltda, 2003. Citations on pages 15, 21, 27, and 28.

RIBEIRO, L. F.; SAVERESE, P. H.; FIGUEIREDO, D. R. struc2vec: Learning node representations from structural identity. In: **Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining**. "": "", 2017. p. 385–394. Citation on page 44.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citation on page 34.

ROSSI, R. G. **Classificação automática de textos por meio de aprendizado de máquina baseado em redes**. Phd Thesis (PhD Thesis) — Universidade de São Paulo, 2016. Citation on page 38.

ROSSI, R. G.; LOPES, A. A.; REZENDE, S. O. A parameter-free label propagation algorithm using bipartite heterogeneous networks for text classification. In: **Proceedings of the 29th annual acm symposium on applied computing**. "": "", 2014. p. 79–84. Citations on pages 21 and 42.

ROSSI, R. G.; LOPES, A. de A.; REZENDE, S. O. Using bipartite heterogeneous networks to speed up inductive semi-supervised learning and improve automatic text categorization. **Knowledge-Based Systems**, Elsevier, v. 132, p. 94–118, 2017. Citation on page 48.

SANH, V.; DEBUT, L.; CHAUMOND, J.; WOLF, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. **arXiv preprint arXiv:1910.01108**, 2019. Citation on page 33.

SANTOS, B. N.; ROSSI, R. G.; REZENDE, S. O.; MARCACINI, R. M. A two-stage regularization framework for heterogeneous event networks. **Pattern Recognition Letters**, Elsevier, v. 138, p. 490–496, 2020. Citations on pages 48, 61, and 72.

SANTOS, B. N. d.; ROSSI, R. G.; MARCACINI, R. M. Transductive event classification through heterogeneous networks. In: **Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web**. "": "", 2017. p. 285–292. Citation on page 59.

SATHYA, R.; ABRAHAM, A. Comparison of supervised and unsupervised learning algorithms for pattern classification. **International Journal of Advanced Research in Artificial Intelligence**, Citeseer, v. 2, n. 2, p. 34–38, 2013. Citation on page 34.

SETTY, V.; HOSE, K. Event2vec: Neural embeddings for news events. In: **The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval**. [S.l.: s.n.], 2018. p. 1013–1016. Citations on pages 22, 48, 60, and 70.

SHI, C.; LI, Y.; ZHANG, J.; SUN, Y.; PHILIP, S. Y. A survey of heterogeneous information network analysis. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 29, n. 1, p. 17–37, 2016. Citations on pages 22, 59, 60, and 70.

TANG, J.; QU, M.; WANG, M.; ZHANG, M.; YAN, J.; MEI, Q. Line: Large-scale information network embedding. In: **Proceedings of the 24th international conference on world wide web**. "": "", 2015. p. 1067–1077. Citation on page 45.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. In: **Advances in neural information processing systems**. "": "", 2017. p. 5998–6008. Citations on pages 15 and 32.

VENTER, M.; STRYDOM, D.; GROVÉ, B. Stochastic efficiency analysis of alternative basic grain marketing strategies. **Agrekon**, Taylor & Francis, v. 52, n. sup1, p. 46–63, 2013. Citation on page 69.

WANG, J.; WANG, Z.; LI, X.; ZHOU, H. Artificial bee colony-based combination approach to forecasting agricultural commodity prices. **International Journal of Forecasting**, Elsevier, 2019. Citation on page 72.

WILLIAMS, R. J.; PENG, J. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. **Neural computation**, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , v. 2, n. 4, p. 490–501, 1990. Citation on page 35.

WU, Z.; PAN, S.; CHEN, F.; LONG, G.; ZHANG, C.; PHILIP, S. Y. A comprehensive survey on graph neural networks. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, 2020. Citations on pages 22, 60, and 70.

XUE, F.; HONG, R.; HE, X.; WANG, J.; QIAN, S.; XU, C. Knowledge based topic model for multi-modal social event analysis. **IEEE Transactions on Multimedia**, IEEE, 2019. Citations on pages 59 and 70.

YANG, C.; ZHANG, J.; HAN, J. Neural embedding propagation on heterogeneous networks. In: IEEE. **2019 IEEE International Conference on Data Mining (ICDM)**. "", 2019. p. 698–707. Citations on pages 22, 34, 47, 53, and 61.

YANG, Y.; LICHTENWALTER, R. N.; CHAWLA, N. V. Evaluating link prediction methods. **Knowledge and Information Systems**, Springer, v. 45, n. 3, p. 751–782, 2015. Citation on page 60.

YU, B.; HU, J.; XIE, Y.; ZHANG, C.; TANG, Z. Rich heterogeneous information preserving network representation learning. **Pattern Recognition**, Elsevier, v. 108, p. 107564, 2020. Citation on page 47.

ZAVERI, A.; KONTOKOSTAS, D.; SHERIF, M. A.; BüHMANN, L.; MORSEY, M.; AUER, S.; LEHMANN, J. User-driven quality evaluation of DBpedia. In: . New York, NY, USA: Association for Computing Machinery, 2013. (I-SEMANTICS '13), p. 97–104. ISBN 9781450319720. Citation on page 81.

ZHANG, C.; SONG, D.; HUANG, C.; SWAMI, A.; CHAWLA, N. V. Heterogeneous graph neural network. In: **Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. [S.l.: s.n.], 2019. p. 793–803. Citation on page 60.

ZHOU, D.; BOUSQUET, O.; LAL, T. N.; WESTON, J.; SCHÖLKOPF, B. Learning with local and global consistency. In: **Advances in neural information processing systems**. "": "", 2004. p. 321–328. Citations on pages 40, 42, 48, 53, and 54.

ZHOU, F.; WU, B.; YANG, Y.; TRAJCEVSKI, G.; ZHANG, K.; ZHONG, T. Vec2link: Unifying heterogeneous data for social link prediction. In: **Proceedings of the 27th ACM International Conference on Information and Knowledge Management**. "": "", 2018. p. 1843–1846. Citation on page 49.

ZHU, X.; GHAHRAMANI, Z.; LAFFERTY, J. D. Semi-supervised learning using gaussian fields and harmonic functions. In: **Proceedings of the 20th International conference on Machine learning (ICML-03)**. "": "", 2003. p. 912–919. Citation on page 39.

ZONG, C.; XIA, R.; ZHANG, J. **Text Data Mining**. [S.l.]: Springer, 2021. Citation on page 29.


ICMC
USP
SÃO CARLOS