UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

Evaluating Pre-trained Word Embeddings in domain specific Ontology Matching

Sofia Pessoa de Amorim

Mestrado em Ciência de Dados

Dissertação orientada por:

Professora Doutora Cátia Luísa Santana Calisto Pesquita

2021

"If you can't fly, then run, if you can't run then walk if you can't walk then crawl, but whatever you do you have to keep moving forward"

— Martin Luther King, Jr.

Acknowledgements

To my supervisor, Professor Cátia, I want to express my sincere gratitude for the patience, encouragement, presence, support and guidance. Her work ethic and dedication to her students are truly inspiring and exceptional.

I also want to thank my Data Science colleagues for making these years special through incredible conversations, fundamental support and endless kindness. Their knowledge, compassion and companionship kept me motivated, humble and curious.

To my parents, my forever gratitude for their unconditional support, for keeping me focused and persevering. Their lessons, positivity and uplift were vital in making the conclusion of this dissertation possible.

To my dearest sisters, a warm and honest thank you for their love and support. Their brilliance, dedication and humbleness are my forever motivation.

Lastly, thank you to my friends and family for always being present and keeping me joyful.

Abstract

The ontology matching process focuses on discovering mappings between two concepts from distinct ontologies, a source and a target. It is a fundamental step when trying to integrate heterogeneous data sources that are described in ontologies. This data represents an even more challenging problem since we are working with complex data as biomedical data. Thus, derived from the necessity of keeping on improving ontology matching techniques, this dissertation focused on implementing a new approach to the AML pipeline to calculate similarities between entities from two distinct ontologies.

For the implementation of this dissertation, we used some of the OAEI tracks, such as Anatomy and LargeBio, to apply a new algorithm and evaluate if it improves AML's results against a reference alignment. This new approach consisted of using pre-trained word embeddings of five different types, BioWordVec Extrinsic, BioWordVec Intrinsic, PubMed+PC, PubMed+PC+Wikipedia and English Wikipedia. These pre-trained word embeddings use a machine learning technique, Word2Vec, and were used in this work since it allows to carry the semantic meaning inherent to the words represented with the corresponding vector. Word embeddings allowed that each concept of each ontology was represented with a corresponding vector to see if, with that information, it was possible to improve how relations between concepts were determined in the AML system. The similarity between concepts was calculated through the cosine distance and the evaluation of the new alignment used the metrics precision recall and F-measure. Although we could not prove that word embeddings improve AML current results, this implementation could be refined, and the technique can be still an option to consider in future work if applied in some other way.

Keywords: Word Embeddings, Ontology Matching, Semantic Web

Resumo Alargado

Desde o surgimento da internet que a quantidade de informação transmitida entre entidades aumentou exponencialmente. Num ambiente em que o processo de produção de informação é exponencialmente superior à capacidade do ser humano reconhecer e assimilar o conhecimento contido nela, tornou-se imperativo que novas estratégias fossem desenvolvidas. A este fator acrescenta também a dificuldade de que os dados têm uma elevada heterogeneidade, pelo facto de o conhecimento ser produzido e transmitido por indivíduos diferentes, e por poder ser definido de diversas maneiras, recorrendo a palavras diferentes. Neste sentido gerou-se o conceito de ”Web Semântica” para potencializar a utilização dos dados em tecnologias e aplicações avançadas. Quando aplicada a dados biomédicos, a tarefa torna-se ainda mais complexa, não só devido à sua contínua e exponencial expansão, levando a um aumento significativo do volume de dados, como também pela complexidade e heterogeneidade da própria informação. Derivado destes fatores, surgiu então a necessidade de organizar e processar esta informação de maneira que pudesse ser interpretada e estudada pelo ser humano.

O conceito de ontologia surge como um documento formal que estrutura e define detalhadamente diferentes conceitos, de um domínio específico, com as respetivas inter-relações. Consequentemente, cria-se uma contextualização e caracterização do domínio, permitindo deste modo a extrapolação da informação para interligação com domínios paralelos. Este paralelismo deve-se a diferentes entidades que produzem ontologias referentes a domínios semelhantes, mas com granularidades diferentes, com recurso a palavras diferentes e/ou de forma descoordenada.

De forma a conseguir integrar o conhecimento contido em duas ontologias distintas, o conceito de alinhamento emergiu. O alinhamento permite encontrar correspondências entre conceitos de ontologias diferentes. Contudo, o procedimento e estratégia que levam a que este alinhamento seja feito da maneira mais precisa e correta requer ainda uma constante melhoria.

Para incentivar o desenvolvimento contínuo de sistemas cada vez mais eficazes na produção de alinhamentos entre ontologias, a Ontology Alignment Evaluation Initiative (OAEI) foi criada em 2004. A OAEI é ainda uma plataforma de referência onde são revelados anualmente os sistemas mais eficazes no alinhamento de ontologias, tais como o Agreement Maker Light (AML), por exemplo. Embora já existam sistemas, como o AML, capazes de encontrar alinhamentos verdadeiros com uma elevada precisão, é ainda preciso uma evolução na maneira de caracterizar os conceitos. Através da similaridade de palavras é possível determinar relações entre palavras que são textualmente semelhantes ou que têm sinónimos que as ligam. Contudo, palavras que partilham semântica, mas não são representadas de forma similar, facilmente são excluídas destes sistemas quando tentamos encontrar conceitos correspondentes aos mesmoss noutras ontologias.

Esta dissertação propõe um novo algoritmo a integrar no AML no qual são utilizados modelos de *embeddings* de palavras pré-treinados para representar diferentes ontologias através de vetores. Os *embeddings* são usados em diferentes *ontology tracks* da OAEI. Com estes *embeddings* integrados nos diferentes pares de ontologias da OAEI, ficamos com um vetor por cada conceito presente em cada ontologia. Deste modo, para conceitos com mais do que uma palavra, aplicou-se o produto Hadamard para que a cada palavra correspondesse apenas um vetor. Como tal, as coordenadas dos vetores de cada palavra pertencente a esse conceito são multiplicadas pelas respetivas, das restantes palavras. Por outro lado, para as palavras que não existiam nos modelos de *embeddings*, aplicou-se um vetor preenchido com 1s, com uma dimensão correspondente ao modelo em que seria aplicado.

Com esta informação discriminada por cada conceito de cada ontologia, procedeu-se à integração da mesma no novo algoritmo que calcula a distância de cossenos para a produção de um novo alinhamento. Esta distância mede-se entre os vetores dos conceitos da ontologia *source* e os da ontologia *target*. Adicionalmente ao matcher que integra este algoritmo, criou-se um matcher complementar que combina o atualmente implementado *String Matcher* (SM) com o novo algoritmo que considera os *embeddings* de palavras, Word Embedding Matcher (WEM). Finalmente, os resultados deste alinhamento são avaliados contra o alinhamento de referência (correspondente ao par de ontologias em causa), com recurso a três métricas diferentes: *precision*, *recall* e *F-measure*.

Em alguns casos, verificou-se que os *embeddings* de palavras tiveram algumas melhorias, especialmente em ontologias de grandes dimensões. Contudo, na generalidade, esta estratégia ficou aquém do esperado uma vez que o *matcher* atualmente implementado (SM) obtém melhores resultados do que o novo *embedding matcher* (WEM).

A estes resultados poder-se-iam implementar algumas melhorias, tais como: realizar *background knowledge*; adquirir maior capacidade de computação, para possibilitar a testagem desta estratégia em ontologias maiores (*e.g.*, SNOMED-NCI); experimentar esta implementação noutros sistemas para além do AML ou experimentar combinar este algoritmo com o AML de outra forma.

Palavras-Chave: *Embeddings* de Palavras, Alinhamento de Ontologias, Ontologias Biomédicas

Contents

List of Figures

List of Tables

Acronyms

NLP	Natural Language Processing
AML	AgreementMakerLight
OAEI	Ontology Alignment Evaluation Initiative
KG	Knowledge graphs
RDF	Resource Description Framework
URI	Uniform Resource Identifier
IRI	Internationalized Resource Identifier
OWL	Web Ontology Language
CBOW	Continuous Bag-Of-Words
SG	Skip-Gram
SCBOW	Siamese Continuous Bag-Of-Words
NCI	National Cancer Institute
LargeBio	Large Biomedical Ontologies
FMA	Foundational Model of Anatomy
MeSH	Medical Subject Headings
PMC	PubMed Central
AML (SM)	Agreement Maker Light String Matcher
AML (WEM)	Agreement Maker Light Word Embedding Matcher
AML (SM+WEM)	Agreement Maker Light String and Word Embedding Matcher
BERT	Bidirectional Encoder Representations from Transformers

Chapter 1

Introduction

Finding ways to aggregate different concepts is a tremendous demanding task for humans. When tackling tasks like these with advanced technologies, such as Artificial Intelligence applied to Big Data, we need to ensure that the data is processed efficiently (with more scalability). Moreover, it should produce viable insights about a particular data set, meaning it processes significant amounts of data even better than humans do. In this sense, several studies have been developed throughout the years to find the best ways to improve how machines process knowledge produced by humans. This knowledge is represented through concepts, and every concept holds a meaning. Additionally, every concept can be described in different ways and with different words. Humans turn concepts into knowledge by taking all concepts made out of information captured by our senses and finding a relation between them, giving them a sense of meaning and a way to transfer this information between individuals.

Ontologies emerged from the necessity of representing various concepts with descriptions of a common subject and a relation explanatory of their connection. Each relation between objects has a particular type or class and can be domain-specific. Relationships are often used to store specific facts or answer particular types of questions. Ontologies are used in several scientific areas to describe and organise a domain. They provide an explicit specification of each entity in a domain, facilitating the standardisation of communication between people, organisations and software systems. By doing so, a shared understanding of concepts is created. For example, suppose we manage to map concepts that belong to two different species in the same ontology. In that case, we could expand our research by understanding how the same concepts (or related) interact in both species.[6, 27]

However, when different ontologies are used to describe different datasets, there is a need to match the ontologies, *i.e.*, to find which elements of an ontology are equivalent. Ontology matching focuses on aligning two different ontologies by finding matching concepts. When comparing two ontologies, the main challenges revolve around two things: the complexity of the data and the words, structure and semantics used to describe each concept between both ontologies. Usually, ontologies have a large and diverse amount of classes, and thus, it can become challenging to work with this data. To overcome the Big Data problem, the information needs to be efficiently allocated into computer memory and, posteriorly, analysed to know how well and with which concept matches better with the ontology we are comparing.

Simultaneously, due to human idiosyncrasy, even if there are analogous concepts between two ontologies, they can be described with different words, creating another challenge to identify their equivalence. Although some ontologies define synonyms and partially help tackle this issue, the mismatch between the ontologies' vocabularies is still the main reason for lower performance in ontology matching. This problem is severe in domains where homonyms and synonyms are very common, which is the case of biomedical ontologies [12].

Essentially, there are many biomedical ontologies that cover the same or intersecting domains, which leads to a significant need on aligning them to integrate the data that is described and contained within these ontologies.

Nevertheless, there are still many possible avenues to search for the best strategy to tackle the problem despite the breakthroughs.

1.1 Objectives

In the last few years, word embedding techniques have been used to represent the semantics of words in a given textual corpus [24]. These techniques represent each word as a numerical vector, and operations over these vectors are used to detect related words.

Word embeddings are often successful in many Natural Language Processing (NLP) tasks [24, 19], but their performance in different domains is affected by the *corpus* where the embeddings were initially trained. This dissertation aims to evaluate the performance of using different pre-trained word embeddings in ontology matching algorithms to align domain-specific ontologies. This study incorporates word-embeddings based ontology matching approaches with AML, a state of the art system for large biomedical ontologies and schema matching. AML is also one of the top-performing systems in the biomedical tracks of the Ontology Alignment Evaluation Initiative (OAEI) since the beginning of its development in 2013 [11]. It uses five different pre-trained word embeddings, trained with different approaches and corpora, and evaluates the ontology matching performance with benchmark data of the OAEI.

1.2 Dissertation Outline

This dissertation is organised into five chapters:

Chapter 1, the current chapter, introduces the context and motivation as well as an overview of the implemented approach to improve AML's pipeline;

Chapter 2 focuses on describing the fundamental concepts inherent to this dissertation as well the related work done so far related to this implementation;

Chapter 3 describes how the developed implementation was constructed as the tools and data used for this dissertation;

Chapter 4 describes and presents the obtained results and a discussion on the main key takeaways based on the final results;

Chapter 5 presents the conclusions that this dissertation led to and what could have been done differently to improve the outcome.

Chapter 2

Background

2.1 Semantic Web

The Semantic Web concept was first described as “not a separate Web but an extension of the current one, in which information is given well-defined meaning, better-enabling computers and people to work in cooperation” [2]. Essentially, the main goal of the Semantic Web is to make machines better at automatically processing and ”understanding” the Internet data, which usually is designed for humans to understand and not for machines. Furthermore, Semantic web aims to make it possible for machines to relate concepts dynamically, identifying the similarity between concepts with different labels and bridging concepts intelligible to both humans and machines.

This necessity of organising information in a way that would be understandable to both researchers and computers triggered the creation of new and more complex representations, such as ontologies.

The concept Resource Description Framework (RDF) emerges in this phase as a foundation for processing metadata, data that describes web resources. These files have proven to be resourceful when describing ontologies, for example [17].

2.1.1 Ontologies

As defined by Studer *et al.* in 1998, “an ontology is a formal, explicit specification of a shared conceptualisation” [26]. They are a means to formally model the structure of a system, meaning the relevant entities and relations that are associated with their observation, and that, in a way, can be helpful to what motivates the study of a subject [25]. Ultimately, ontologies are used as the schema layer of KGs. Ontologies were developed to help professionals describe the various relationships between distinct entities. They provide a vocabulary that describes a domain and a specification of the meaning intrinsic to the terms present in that vocabulary. By conceptualising what these vocabulary terms express in a computer format, ontologies become critical components of the Semantic Web [7]. Furthermore, they become valuable assets to represent big and complex domains by improving machine understanding and decreasing ambiguity.

Ontologies can be populated with individuals, *i.e.* instances or objects at the data level, and can contain literals represented by an integer or, most commonly, a string. They are organised in different classes that can be defined as categories or collections of individuals (concepts) and properties. Essentially, classes allow the creation of a relation between an individual and another individual or a literal, enabling the creation of hierarchies of classes and properties constituted by subclasses and sub-properties. If an individual has an Internationalized Resource Identifier (IRI), which is similar to an URI but with an extended character set, to identify it precisely, it is called an entity (or named individual).

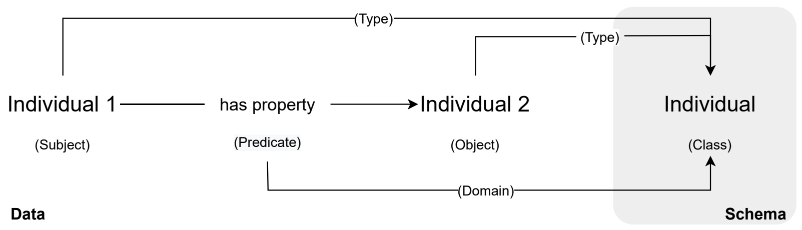

Figure 2.1: Example of an ontology statement

By doing so, computer applications are more efficient at analysing, matching and reason about complex knowledge, automatically [8].

For a computer to represent relationships between entities and terms, including more detailed properties, semantic meaning, and knowledge consistency, ontologies are commonly represented in a more expressive language than RDF, such as Web Ontology Language (OWL). It is possible to structure and organise all the relevant and complex knowledge with these documents.

2.1.2 Ontology Matching

Ontology Matching focuses mainly on aligning, as correctly as possible, two or more ontologies, that cover the same or intersecting domains, so that the knowledge contained in different ontologies can be combined, to expand our knowledge about that domain and create a shared understanding about concepts. In theory, each concept should match only one definition. An ontology should be built in such a broad and specific way that it should not overlap content with other ontologies. Nevertheless, because it is an active topic of research, it is common for a domain to be represented by various ontologies. The leading cause for this problem is the heterogeneity between descriptions of one single concept, which is caused by the creation of ontologies in an uncoordinated way. Different individuals use different names and can use a more advanced or basic lexicon to describe the same concept, making the task of finding common ground between the data from these ontologies across the Semantic Web a much more complex challenge. Ontology alignment/matching arises as a solution to this problem as it focuses on finding the most accurate correspondence between concepts of two different ontologies, source and target.

Fundamentally, Ontology matching is the process that makes it possible to find connections between concepts of distinct ontologies [7] so all the knowledge contained in multiple ontologies can be related and augmented.

2.1.3 Ontology Alignment Evaluation Initiative

The Ontology Alignment Evaluation Initiative [1], created in 2004, occurs every year to test and evaluate how state-of-the-art matching systems perform on complex matching tasks to keep an updated ranked list to inform which tools are improving at these challenges. In addition, this initiative evaluates the performance of the selected matching systems with biomedical ontologies of considerable size.

2.1.4 AgreementMakerLight

AgreementMakerLight is a system created in Java to improve the ontology matching process and results [2]. The development of AML was derived from Agreement Maker, one of the first reference ontology matching systems[4]. However, these primordial systems did not consider scalability as much as large ontologies needed to provide efficiency. As an automated system, AML added the required scalability while still preserving the advanced flexibility and extensibility that AgreementMaker provided [11].

AML is structured with three main modules, each with a specific role. The pipeline starts with the loading module, in which ontologies in files of type OWL are loaded into memory objects with hash-based data structures (*Lexicon* and *RelationshipMap*). This indexing enables an optimisation of the memory space and makes retrieving the data faster and more efficient. The *Lexicon*, a table of class names, labels and synonyms of each concept inside an ontology, uses a ranking system to weight them and score their matches [5], while *RelationshipMap* stores relations between the concepts that are within *Lexicon*. At the end of this module, both ontologies to be matched, source and target, should be loaded into AML. An additional ontology can be loaded to establish background knowledge bridges between the input ontologies. This background knowledge could also be an alignment (in RDF format), used as a reference alignment, which should be a good evaluator of the alignment or a good baseline for an extension of a new alignment. It could minimise the number of matches to be computed and reduce the computational workload, increasing the performance.

The second module corresponds to the matching module. This module represents the most relevant part of the pipeline since it includes all matching algorithms and strategies, organised in classes, that

[1]http://oaei.ontologymatching.org
[2]Code and data available at https://github.com/AgreementMakerLight/AML-Project.git

make this a state of the art system for ontology matching. These classes, the *Matchers*, are responsible for producing the object that constitutes the alignment, where all correspondences between the source and target ontologies entities are determined. The set of mappings created in this alignment have a calculated similarity for each pairing and are filtered by a predefined threshold. However, this similitude between every concept (from a source ontology) against each concept of the target ontology constitutes a $O(n^2)$ complexity. Thus, to preserve efficiency, the matching algorithms are distributed into primary and secondary matchers.

Primary matchers are focused on HashMap cross-searches, making these matchers reach a $O(n)$ complexity while searching for matching possible combinations. For example, the Lexical Matcher is one of the primary matchers in AML since it bases the search on identifying entities with similar names. The secondary matchers compare each entity of the source ontology with all of the target ontology entities, increasing the search's complexity to $O(n^2)$. Therefore, when using these matchers in large ontologies, it is crucial to use a reference alignment to extend the alignment and decrease the computational workload by exploring the neighbourhood of predetermined mapped entities. This extension procedure creates then all possible correspondences between the source and target entities that do not create incompatibilities with the base alignment and finally applies the matching algorithm that calculates the similarity between those pairs. The alignment extension is repeated as many times as needed until there are no more possible combinations. The String Matcher is an example of this second type of matcher since it correlates concepts by string similarity between the *Lexicon* entries of both entities. AML is also a system prepared to run mapping tasks in parallel by using multiple cores from CPU implementations, increasing the system's performance. The third and final module filters the alignment by removing some mappings. The removal focuses on cardinality and logical issues. The cardinality problem arises when multiple relations are associated with a class when only a relation 1:1 should exist. Logical conflicts occur when at least two correspondences are not coherent when merging the ontologies with those mappings. This coherency is significantly relevant, so there are no violations of restrictions of the ontologies. To solve this, AML includes an algorithm, Selector, that removes concurrent mappings by excluding those with less similarity, achieving the appropriate one-to-one cardinality. The selection can be of three types: Strict, Permissive and Hybrid. The first type of selection does not allow any conflicts in the alignment. In contrast, the second one only allows competitive mappings alignments with the same similarity and the last type, hybrid, only allows a cardinality of two between mappings with similarity above 0.75. AML also contains an algorithm that searches for logical incoherences, the Repairer algorithm, and removes or modifies them according to a set of predefined rules [23].

The AML system can be executed with many different tools. However, since it is written in Java, the code needs to be compiled every time we want to run a class. In this sense, the Eclipse project reveals to be a handy tool to run AML since it provides an environment capable and prepared to compile and run the code in a much simpler and faster way. Nevertheless, it could also be executed with its JAR file on a command line or with GUI, the latter being more suitable for less experienced users.

In 2020, AML achieved the highest F-score in an unannounced task that consisted of matching DBpedia to the OntoFarm ontologies out of only five systems that were eligible to participate in it. Overall, AML remains among the top-performing systems in most OAEI tracks and datasets that have participated in the last few years [20]. The overall described AML pipeline is represented in Figure 2.2:

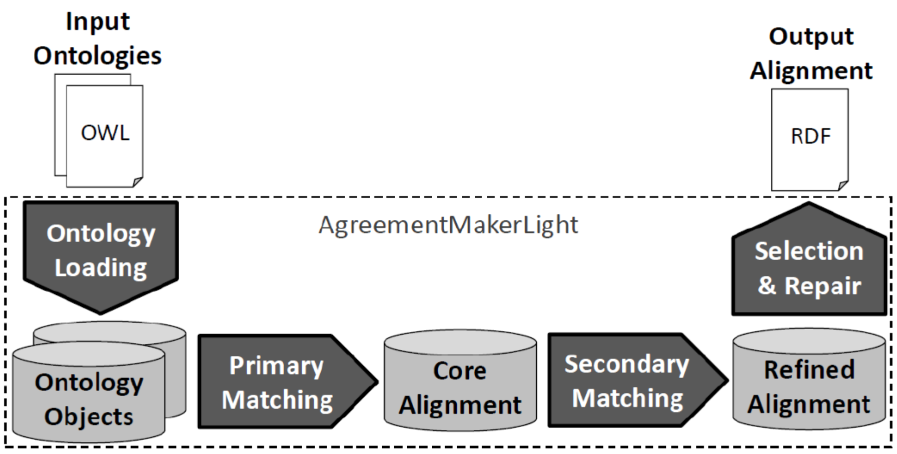

Figure 2.2: AML pipeline. Extracted from Faria *et al.* [10];

Faria *et al.*[11] demonstrated how the linear complexity of the hash-based searching strategy is fundamental when dealing with large biomedical ontologies to do ontology matching with the use of the AML system. The authors show that the results improve significantly by weighting the *Lexicon* (*e.g.* by using concept synonyms to find similarity between concepts), taking into account the lexical annotations and different levels of precision of the different types of synonyms.

Moreover, AML's local string matching strategy also proves to be an effective search space reduction strategy, which decreases computational costs, an essential factor when working with large ontologies [9, 11]. It is also demonstrated that the challenges that ontology matching tasks unfold need systems capable of combining various strategies into a mature matching approach.

Furthermore, the ontologies are parsed and loaded into an alignment system.

At this stage, matching algorithms generate mapping candidates throughout multiple phases and filtering, and to which a score will be assigned (depending on how good the relationship established is). Finally, problem-causing mappings are removed [11].

2.2 Word Embeddings

Word embeddings are vector representations of words given their co-occurrence in a corpus. This representation aim to mirror the human intuitions about similarity and relatedness when analysing different concepts [15]. So, when analysing ontologies we search for related concepts, not necessarily similar ones. This practice has shown to be a good resource in the biomedical domain while tackling these types of problems [1, 24].

Essentially, a neural embedding model is a predictive learning-based model that uses a neural network to embed categorical variables in n-dimensional vectors. This model's learning process uses prediction in an unsupervised way since there is no need for labelling the data.

This technique is mainly used since it reduces the dimensionality of complex data. Doing so also significantly decreases the computational costs while still maintaining an efficient way to represent complex concepts.

It is necessary to use a metric representative of the distance between these two concepts to calculate the similarity. The most common and more efficient metrics used are, *e.g.* the Euclidean, Manhattan, Cosine, Dice and Jaccard [14, 13]. The Cosine distance is considered one of the best metrics when dealing with NLP tasks. The distance between two vectors is measured by calculating the angle between them in their intersection point. It is possible to determine the angle value by doing the dot product between the two vectors, normalised. For example, considering $\vec{v}\ and\ \vec{w}$ as two vectors, representative of two distinct concepts, it is possible to calculate the distance between them with the following formula:

$$cosine(\vec{v}, \vec{w}) = \frac{\vec{v} \bullet \vec{w}}{\mid \vec{v} \mid\mid \vec{w} \mid} = \frac{\sum_{i=1}^{n} v_i \times w_i}{\sqrt{\sum_{i=1}^{n} v_i^2} \sqrt{\sum_{i=1}^{n} w_i^2}}$$

Calculating the similarity between vectors with the cosine similarity implies that the higher the cosine is, the higher the angle between the vectors is and, therefore, the less similar the concepts are.

2.2.1 Word2Vec

The Word2Vec model, introduced by Mikolov *et al.* constituted a baseline for NLP tasks due to its noteworthy results for carrying semantic meaning when using models trained with vector representations of words [21, 22]. With this technique the words are represented with a numerical vector with a pre-defined number of dimensions. In 2013, Mikolov *et al.* developed Word2Vec as a tool, with an efficient implementation of the Continuous Bag-Of-Words (CBOW) and Skip-Gram (SG) architectures, for computing vector representations of words [21, 22]. Two of the factors that made Word2Vec so successful, besides achieving state-of-the-art results on many NLP tasks [18, 19], was its simplicity and accessibility [3].

What distinguishes the CBOW and SG models is that, given a word in the *corpus*, CBOW tries to predict the current word based on the previous and the posterior. On the other hand, SG uses the current word to predict others, previous or next words.

The contrasted way of how both models work is illustrated in the figure 2.3 below :

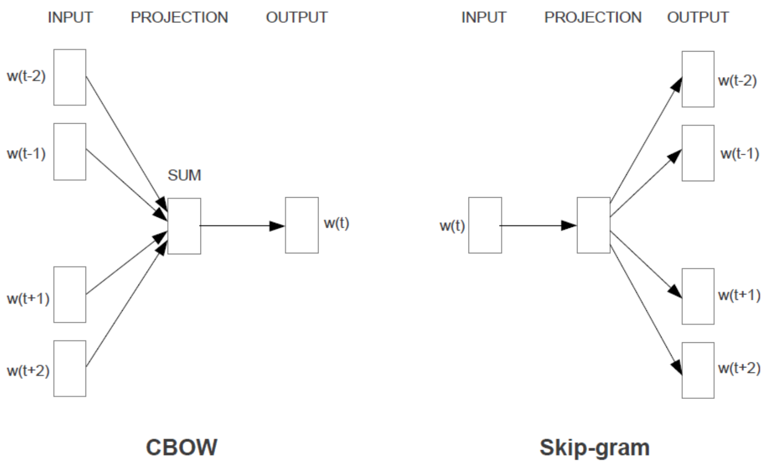

Figure 2.3: On the left side of the figure is represented the continuous bag-of-word model. Simultaneously, on the right side, the skip-gram model. Extracted from Mikolov *et al.* [21]

The idea behind this technique is that the resultant vector space allows for concepts represented by vectors to be added and/or subtracted intuitively, carrying, therefore, its semantic meaning, as also shown in Figure **??**.

However, concepts frequently have more than one label in the biomedical domain, such as "respiratory system". To solve this problem, Word2Vec emerges as an innovative tool that allows the representation of a concept as a single vector that can then be compared to another one by considering the concept as the mean vector of the word vectors composing the concept. Furthermore, if the concepts are built similarly to a sentence, this average can be optimised by assigning weights to each word. The score of the similarity between these concepts is finally calculated with the cosine distance between the produced vectors, meaning that the vectors with less distance are the ones that are less likely to be related. By creating Word2Vec, Mikolov *et al.* aimed to prove that it was possible to obtain vector representations with high quality even if these models had a very simple architecture [21]. A simple architecture represents a significant factor when working with large datasets, where the computational workload increases significantly.

2.2.2 State-of-the-art

After Word2Vec was proposed in 2013, Kolyvakis *et al.*[16] presented *Deep Align* in 2018. *Deep Align* is based on the Siamese CBOW (SCBOW), which is an extension of CBOW, as an ontology matching framework, using supervised learning to predict consecutive sentences while optimising pre-trained word embeddings to generate improved sentence embeddings.

The framework introduced by the authors is composed of two neural network components that learn which alignments are semantically similar. The first component finds a large set of possible accurate mappings between two ontologies, and the second makes a more profound analysis of these alignments to correct errors. These alignments are built according to the cosine similarity between each entity from the source target ontology. Then, the source alignment is compared to every entity of the target ontology. The concepts with smaller cosine distances are removed, and only the rest are kept. SCBOW rewards sentences that frequently appear together while penalises sentences that do not. SCBOW also uses a DAE, a denoising autoencoder, to find errors in the alignments through a mechanism that identifies outliers. This procedure is essential since it captures intrinsic characteristics of terms with similar semantics to exclude related terms. Moreover, this approach could be an interesting strategy in ontology alignment tasks.

Even though this technique had promising results with some OAEI biomedical tasks, it needs labelled data since it is a supervised learning method. Thus, the labelled data is dependent on having truly correct alignments. In addition, they are used as training data, constituting a complex challenge when working with the biomedical domain, where this information is still sparse. However, Kolyvakis *et al.* only considered leading names from concepts while AML, for instance, takes into consideration lexical annotations.

Driven by the motivation of creating a unified knowledge structure, Li G. suggested in 2020 a method that combines the representation learning method as a component with traditional feature engineering methods to improve matching system's performance [19]. The author suggests word embeddings as a representation learning method to be able to carry semantic information contained in their labels. The implementation of Li G. based on implementing word embeddings, pre-trained with PubMed, PMC texts and English Wikipedia dump, in existing matching systems such as LogMap, FCAMapX and FCAMap-KG, ALIN and Lily. The data set used for this work purpose was the Adult Mouse Anatomy as source and National Cancer Institute Thesaurus (NCI) as target ontology. In conclusion, the strategy of using word embeddings in the previously mentioned matching systems showed that although precision decreases a little when using word embeddings, recall can increase in most cases. Thus, using word embeddings in systems like these seem to improve ontology matching systems results.

Chapter 3

Methodology

3.1 Overview

The data used for this dissertation focused on combining different ontologies and word embeddings techniques to test if they were representative that word vectors would improve AML performance in different ontology matching tasks or not. Furthermore, a reference alignment was used for each ontology pair to establish a common ground to evaluate if these approaches were reliable or not.

The utilised datasets were extracted from portions of the Ontology Alignment Evaluation Initiative 2021 tracks:“Anatomy” and “Large Biomedical Ontologies (LargeBio)”.

In the first stage, the source and target ontologies’ concepts names were separately extracted with a class from AML into two different text files with all respective names listed.

The next phase focused on integrating and associating vectors coordinates to each ontology concept, based on different types of pre-trained word embeddings. This stage was performed in python, as described in 3.3. When concepts were described with more than one word, each coordinate of each word was aggregated through the Hadamard Product by multiplying with the other corresponding coordinates. For words that were present in the ontologies but did not exist in these word embeddings models, was given a vector with an n dimension with all coordinates set to ”1” so that they would not influence the aggregated vector (since the number ”1” has no impact in a multiplication result). The number of cases to which there was no corresponding word in the word embedding model was also recorded and it is presented in tables 4.1, 4.2, 4.3, 4.4 and 4.5. After having all concepts names in one column and the corresponding vector inside a data frame, the data was exported to a CSV file to be then imported to AML, along with the respective ontologies.

With both source and target concepts (separately) associated with a vector, the data was reinserted in AML to create a new alignment. All evaluations were computed with the Background Knowledge matchers turned off. Background knowledge matchers explore other ontologies and controlled vocabularies to find additional mappings. These external resources are seldom available in real-world scenarios, so to better highlight the contributions of the word embeddings to the performance, these matchers were removed.

In this dissertation, three different settings of AML were tested: (1) regular AML, with String Matcher (SM); (2) a new version of AML that replaces the String Matcher method (based on string similarity metrics) with the Word Embedding Matcher (WEM) that calculates the cosine similarity between concepts and (3) one that combines the String Matcher and the Word Embedding Matcher (SM+WEM).

In the last phase of this dissertation's pipeline the newly produced alignment is evaluated against the reference alignment, returning the precision, recall and f1-measure. This pipeline is represented in the figure 3.1 below:

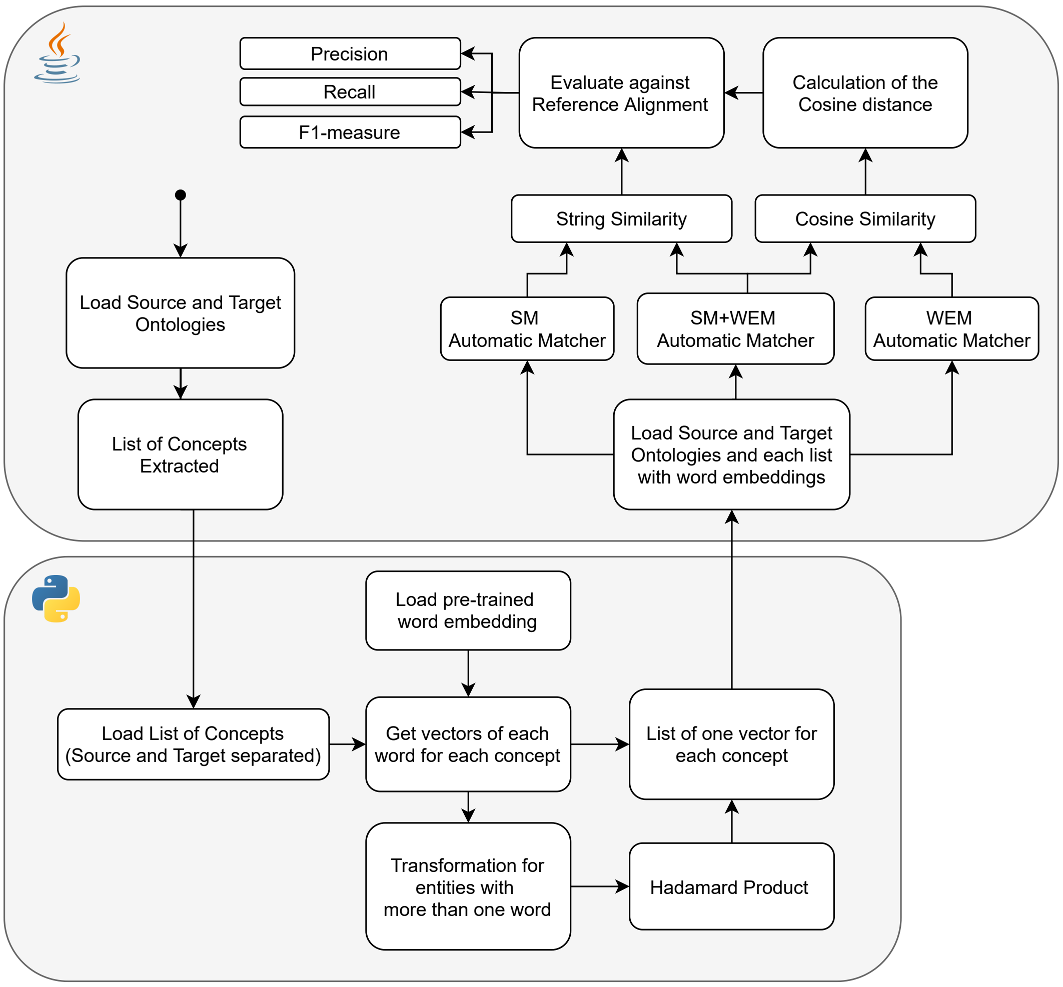
Precision
Recall
F1-measure
Evaluate against Reference Alignment
Calculation of the Cosine distance
String Similarity
Cosine Similarity
Load Source and Target Ontologies
SM Automatic Matcher
SM+WEM Automatic Matcher
WEM Automatic Matcher
List of Concepts Extracted
Load Source and Target Ontologies and each list with word embeddings
Load pre-trained word embedding
Load List of Concepts (Source and Target separated)
Get vectors of each word for each concept
List of one vector for each concept
Transformation for entities with more than one word
Hadamard Product

Figure 3.1: Dissertation Pipeline

3.2 Ontologies and Reference Alignments

To evaluate if pre-trained word embeddings improved the state-of-the-art ontology matching AML tool, five pairs of ontologies were tested. The ontologies used are presented in Table 3.1:

Source	**Target**	**Total number of Concepts (Source)**	**Total number of Concepts (Target)**	**Total number of mappings (Reference)**
Mouse	Human	3072	5074	1516
small FMA	NCI	11515	19515	3024
large FMA	NCI	222991	190743	
small FMA	SNOMED	36473	26295	9008
large FMA	SNOMED	222983	184467	

Table 3.1: Source and Target ontologies extracted from OAEI tracks, corresponding number of concepts and reference mappings

The Anatomy track focuses on aligning the adult Mouse anatomy with a portion of the NCI Thesaurus that describes the human anatomy. These ontologies differ since they widely use the partOf relation between the constituent entities. On the other hand, the LargeBio ontology track will be used to produce alignments between SNOMED CT ontology, the Foundational Model of Anatomy (FMA) and lastly, the NCI. The latter track is very semantically rich and contains tens of thousands of classes. For this reason, the LargeBio track datasets are much heavier to work with than the mouse and human ontologies. Consequently, we could not make all the possible combinations (such as small/large SNOMED with NCI) since it turned out to be too heavy to compute. In terms of memory space allocated to perform these tasks, 11GB were made available only to execute AML.

3.3 Pre-trained Word Embeddings

After extracting all concepts names from the source and target ontologies, five different pre-trained word embeddings techniques were applied to each pair of ontologies.

The combination between concepts names and a corresponding vector was processed in python. The loading and processing of the pre-trained word embeddings binary files was performed with Gensim's library. Gensim is an open-source library mainly used for NLP and information retrieval tasks. It is primarily used for complex tasks, like the ones previously mentioned, due to its efficient multicore implementations of high-performance algorithms. It also does not load everything directly into memory, becoming independent of the size of the *corpus*.

After loading the pre-trained word embeddings into python, we applied a function that takes the text file with all concepts names extracted from AML as an input. Then, it searches in the pre-trained word embeddings for the corresponding vectors for each word in each line. When a concept had more than one word, the corresponding vectors were aggregated with the Hadamard product into a single one. The Hadamard product consisted of multiplying each word's coordinates with the ones of the remaining words of each concept. However, some of the ontologies' concepts were not present in the word embeddings, so, to overcome this problem, to these words, we created a vector with the number ”1” (one) for every coordinate. The dimension of this vector would depend on the applied word embedding technique. Having a vector filled with ones, the result of the product between the word vectors would not influence the final vector.

The used word embedding techniques are listed in Table 3.2 below:

Source- Target Ontologies	BioWordVec Extrinsic: vectors with 200 dimensions;
	BioWordVec Intrinsic: vectors with 200 dimensions;
	PubMed+PC: vectors with 200 dimensions;
	Wikipedia+PubMed+PC: vectors with 200 dimensions;
	English Wikipedia: vectors with 300 dimensions.

Table 3.2: Word Embedding techniques applied to each source and target ontology

All files were extracted from their source as binary files to be then imported with the Gensim library.

BioWordVec [1] is an open set of biomedical word embeddings that merges sub-word information from unlabelled biomedical text with the Medical Subject Headings (MeSH), a commonly used biomedical ontology. The sub-words consist of dividing one word into fragments to better understand and find relations with other words. This dataset is separated into two files: the BioWordVec Extrinsic and the BioWordVec Intrinsic. The first embedding file focuses on extrinsic tasks, useful for relation extraction or text classification and the latter, intrinsic, is more oriented to calculate or predict semantic similarity between words. Both files contain word vectors with a dimension equal to 200. These files contain more than two thousand different words, where approximately 99% of them are from PubMed and the remaining 1% from MeSH.

The PubMed+PC word vectors were induced by all publication abstracts from PubMed and all full-text documents from PubMed Central Open Access subset (PMC), creating a sizeable *corpus* of biomedical and general-domain scientific information combined with the Word2Vec tool. The Wikipedia+PubMed+PC is similar to the PubMed+PC but with the addition of a recent English Wikipedia dump [2].

Lastly, the English Wikipedia was extracted from Wikipedia2Vec [3]. Unlike the previous pre-trained

[1]Files available at [28]
[2]Files available at http://evexdb.org/pmresources/vec-space-models/.
[3]Files available at https://wikipedia2vec.github.io/wikipedia2vec/pretrained/

word embeddings sets, the English Wikipedia included vectors with 300 dimensions instead of 200 and had a much bigger coverage (comparing with the other models).

3.4 AgreementMakerLight Integration

Finally, for each source and target ontology pair and each pre-trained word embedding, three types of alignments were produced:

Source - Target Ontologies	Word Vector Embedding	AML SM
		AML WEM
		AML SM+WEM

Table 3.3: Different Matchers used. The AML SM represents the String Similarity Matcher, the AML WEM the Word Embedding Matcher and, finally, AML SM+WEM a combination of both Similarity and Word Embeddings Matcher

The auto matcher already implemented in AML, AML SM, works with String Similarity, where the similarity between words is measured through distances between each character of each word. Thus, the function used by AML SM is constituted by different possible string metrics, such as the Levenshtein distance, Jaro–Winkler distance and Q-grams distance.

The algorithm introduced for this dissertation purpose was the Word Embedding Matcher, AML WEM. This algorithm functions with the vectors initially produced in python. It takes the vectors of both source and target ontologies and calculates the distances between the concepts through the cosine distance. As previously mentioned in section 2.2, the cosine similarity measures the angle between two vectors, in this case, a source and target concept, at their intersection point. This angle is calculated by performing the dot product between the two normalised vectors. This calculation was achieved by integrating into the WEM the code described in Algorithm 1, which had a similar structure to the SM but without the string similarity part:

Algorithm 1: Cosine Distance Calculation: this algorithm takes in the source and the target vector array as an input. Then, with a for loop, iterates through each coordinate of the source and the target vector arrays to firstly calculate the dot product between each coordinate of each ontology and store the cumulative results into the *Dot Product* variable. Still inside of the for loop, the normalised vectors are stored in each *Normalised Source vector* and *Normalised Target vector* as each coordinate of the corresponding ontology to the power of two. Finally, the cosine distance is calculated by dividing the dot product variable by the product of the normalised vectors of the source and the target ontologies

```
Input: $Source\ Vector\ Array,\ Target\ Vector\ Array$ ;
$Dot\ Product = 0$;
$Normalised\ Source\ vector\ =\ 0$;
$Normalised\ Target\ vector\ =\ 0$;
/* for loop to iterate each concept vector coordinate inside $Source$ and
   $Target$ list of vectors                                              */
for each coordinate do
    $Dot\ Product\ +=\ Source\ Vector\ coordinate \times Target\ Vector\ coordinate$;
    $Normalised\ Source\ vector\ +=\ SourceVectorcoordinate^2$;
    $Normalised\ Target\ vector\ +=\ Target\ Vector\ coordinate^2$;
end
return $Dot\ Product \div (\sqrt{Normalised\ Source\ vector} \times \sqrt{Normalised\ Target\ vector})$;
```

This algorithm returns all distances between the Source and Target ontologies' concepts. With these distances calculated, it is possible to understand which concepts are most likely to be related to two different ontologies.

The third tested matcher was the AML SM+WEM. This matcher is a combination of the string matcher and the word embedding matcher. The AML SM+WEM has the same structure as the string matcher, but both algorithms are used to extend the alignment.

3.5 Evaluation

To assess this dissertation strategy in applying word embeddings to create alignments in AML, we evaluated the new alignment by comparing it with the reference alignment, a ground-truth alignment.

As mentioned at the beginning of this chapter, the metrics used to evaluate the performance of the WEM, SM and SM+WEM, were Precision, Recall and F1-measure. The precision can be described with the following formula:

$$Precision = \frac{Number\ correct\ mappings}{Total\ number\ of\ mappings\ in\ the\ produced\ alignment}$$

Recall works like precision, but it is applied to the reference alignment:

$$Recall = \frac{Number\ correct\ mappings}{Total\ number\ of\ mappings\ in\ the\ reference\ alignment}$$

Lastly, the F-measure demonstrates the harmonic mean of precision and recall, *i.e.*:

$$Fmeasure = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Chapter 4

Results and Discussion

To evaluate how the new automatic embedding matcher performed in some of the OAEI tasks, we used three metrics: precision, recall and F1-measure. These results are segregated by ontology pairs in tables 4.1, 4.2, 4.3, 4.4 and 4.5.The results were obtained when AML evaluated the newly created alignment against the reference alignment, for each task. The *Nulls* column on these tables represents the number of words that were set to have all coordinates equal to 1 because they were not present in the pre-trained word embedding model.

Pre-trained embedding	Matcher	Precision	Recall	F-measure	Found	Correct	Reference	Nulls
Not-Applicable	1. AML (SM)	**96,10%**	83,60%	89,40%	1319	1267	1516	N/A
1. BioWordVec intrinsic (200d)	2. AML (WEM)	94,60%	81,70%	87,70%	1308	1238	1516	73
	3. AML (SM+WEM)	94,40%	83,80%	88,80%	1347	1271	1516	
2. BioWordVec extrinsic (200d)	2. AML (WEM)	96,00%	81,40%	88,10%	1285	1234	1516	73
	3. AML (SM+WEM)	95,70%	**84,00%**	**89,50%**	1330	1273	1516	
3. PubMed+PC (200d)	2. AML (WEM)	95,60%	80,30%	87,30%	1274	1218	1516	161
	3. AML (SM+WEM)	95,30%	83,80%	89,20%	1332	1270	1516	
4. PubMed+PC+Wikipedia (200d)	2. AML (WEM)	**96,10%**	79,90%	87,20%	1260	1211	1516	150
	3. AML (SM+WEM)	95,80%	83,70%	89,40%	1324	1269	1516	
5. English wikipedia (300d)	2. AML (WEM)	95,80%	80,40%	87,40%	1272	1219	1516	259
	3. AML (SM+WEM)	95,60%	83,80%	89,30%	1330	1271	1516	

Table 4.1: Results of produced alignments for **Mouse** as source and **Human** as target ontology, the pre-trained embedding type and matcher. SM corresponds to String Matcher, WEM to Word Embeddings Matcher, and finally, SM+WEM corresponds to the combination of String and Word Embeddings Matcher.

Regarding the Mouse and Human alignment, we can see that by applying the Word Embeddings Matcher (WEM), the precision, recall and F-measure decrease compared to the String Matcher (SM) results. However, when combining String and Word Embeddings Matcher (SM+WEM), recall increases in all cases, and the F-measure remains the same in one case and increases in another.

Pre-trained embedding	Matcher	Precision	Recall	F-measure	Found	Correct	Reference	Nulls
Not-Applicable	1. AML (SM)	**96,80%**	86,80%	**91,50%**	2409	2331	2686	N/A
1. BioWordVec intrinsic (200d)	2. AML (WEM)	95,20%	86,00%	90,40%	2427	2310	2686	682
	3. AML (SM+WEM)	95,30%	86,80%	90,80%	2448	2332	2686	
2. BioWordVec extrinsic (200d)	2. AML (WEM)	96,00%	85,90%	90,70%	2401	2306	2686	682
	3. AML (SM+WEM)	96,10%	86,80%	91,20%	2426	2331	2686	
3. PubMed+PC (200d)	2. AML (WEM)	96,40%	85,70%	90,70%	2386	2301	2686	1258
	3. AML (SM+WEM)	96,30%	86,80%	91,30%	2421	2331	2686	
4. PubMed+PC+Wikipedia (200d)	2. AML (WEM)	96,40%	85,70%	90,80%	2388	2303	2686	1221
	3. AML (SM+WEM)	96,20%	86,80%	91,30%	2422	2331	2686	
5. English wikipedia (300d)	2. AML (WEM)	96,40%	85,70%	90,80%	2387	2302	2686	2004
	3. AML (SM+WEM)	96,30%	**86,90%**	91,30%	2422	2333	2686	

Table 4.2: Results of produced alignments with **small FMA** as source and **NCI** as target ontology, the pre-trained embedding type and matcher. SM corresponds to String Matcher, WEM to Word Embeddings Matcher, and finally, SM+WEM corresponds to the combination of String and Word Embeddings Matcher.

Concerning the small FMA with NCI alignment, it is clear that the WEM does not improve either precision, recall or F-measure. Nevertheless, when combining both strategies, for SM+WEM results, the precision still decreases, but the recall remains the same in most cases and improves in one. In this last matcher, the F-measure also slightly decreases.

Pre-trained embedding	Matcher	Precision	Recall	F-measure	Found	Correct	Reference	Nulls
Not-Applicable	1. AML (SM)	86,20%	83,00%	84,60%	2586	2229	2686	N/A
1. BioWordVec intrinsic (200d)	2. AML (WEM)	78,40%	81,50%	80,00%	2792	2190	2686	23574
	3. AML (SM+WEM)	76,60%	82,50%	79,40%	2894	2216	2686	
2. BioWordVec extrinsic (200d)	2. AML (WEM)	83,70%	81,80%	82,80%	2626	2198	2686	23574
	3. AML (SM+WEM)	80,60%	82,90%	81,70%	2763	2227	2686	
3. PubMed+PC (200d)	2. AML (WEM)	86,60%	81,50%	84,00%	2527	2189	2686	52289
	3. AML (SM+WEM)	82,90%	82,90%	82,90%	2686	2227	2686	
4. PubMed+PC+Wikipedia (200d)	2. AML (WEM)	**87,80%**	81,90%	**84,80%**	2505	2200	2686	51017
	3. AML (SM+WEM)	83,60%	**83,10%**	83,40%	2671	2233	2686	
5. English wikipedia (300d)	2. AML (WEM)	85,50%	81,50%	83,50%	2562	2190	2686	53600
	3. AML (SM+WEM)	81,80%	83,00%	82,40%	2726	2229	2686	

Table 4.3: Results of produced alignments with **large FMA** as source and **NCI** as target ontology, the pre-trained embedding type and matcher. SM corresponds to String Matcher, WEM to Word Embeddings Matcher, and finally, SM+WEM corresponds to the combination of String and Word Embeddings Matcher.

The alignment produced by large FMA and NCI shows that the precision increases in two cases and decreases in three. Recall and F-measure also decrease except for one case where the F-measure does improve. When the SM+WEM matcher is applied to match these ontologies, the three metrics generally decrease in most cases, with recall improving in one case and remaining the same in another.

Pre-trained embedding	Matcher	Precision	Recall	F-measure	Found	Correct	Reference	Nulls
Not-Applicable	1. AML (SM)	**92,80%**	74,00%	**82,30%**	4801	4457	6026	N/A
1. BioWordVec intrinsic (200d)	2. AML (WEM)	86,70%	73,10%	79,30%	5078	4403	6026	2440
	3. AML (SM+WEM)	86,90%	**74,30%**	80,10%	5153	4476	6026	
2. BioWordVec extrinsic (200d)	2. AML (WEM)	90,10%	73,10%	80,70%	4888	4403	6026	2444
	3. AML (SM+WEM)	90,10%	74,30%	81,50%	4967	4477	6026	
3. PubMed+PC (200d)	2. AML (WEM)	92,60%	72,30%	81,20%	4706	4356	6026	2772
	3. AML (SM+WEM)	92,00%	74,20%	82,10%	4859	4469	6026	
4. PubMed+PC+Wikipedia (200d)	2. AML (WEM)	92,50%	72,30%	81,20%	4709	4358	6026	2667
	3. AML (SM+WEM)	92,00%	74,10%	82,10%	4854	4467	6026	
5. English wikipedia (300d)	2. AML (WEM)	92,10%	72,20%	80,90%	4720	4349	6026	5732
	3. AML (SM+WEM)	91,70%	74,20%	82,00%	4872	4469	6026	

Table 4.4: Results of produced alignments with **small FMA** as source and **SNOMED** as target ontology, the pre-trained embedding type and matcher. SM corresponds to String Matcher, WEM to Word Embeddings Matcher, and finally, SM+WEM corresponds to the combination of String and Word Embeddings Matcher.

In the small FMA with SNOMED alignment, the WEM does not improve either one of the three used metrics. However, for SM+WEM, the precision decreases in all cases, the recall improves in all cases, but the F-measure still narrowly decreases.

Pre-trained word embedding	Matcher	Precision	Recall	F-measure	Found	Correct	Reference	Nulls
Not-Applicable	1. AML (SM)	69,60%	66,70%	68,10%	5777	4018	6026	N/A
1. BioWordVec intrinsic (200d)	2. AML (WEM)	46,60%	64,60%	54,10%	8362	3894	6026	10846
	3. AML (SM+WEM)	45,60%	**67,10%**	54,30%	8875	4044	6026	
2. BioWordVec extrinsic (200d)	2. AML (WEM)	**94,60%**	62,00%	**74,90%**	3954	3739	6026	10846
	3. AML (SM+WEM)	69,60%	66,70%	68,10%	5777	4018	6026	
3. PubMed+PC (200d)	2. AML (WEM)	78,90%	63,50%	70,30%	4852	3826	6026	16768
	3. AML (SM+WEM)	64,10%	66,90%	65,50%	6288	4031	6026	
4. PubMed+PC+Wikipedia (200d)	2. AML (WEM)	79,80%	63,60%	70,80%	4802	3834	6026	15986
	3. AML (SM+WEM)	65,10%	66,90%	66,00%	6190	4032	6026	
5. English wikipedia (300d)	2. AML (WEM)	79,60%	63,40%	70,60%	4800	3821	6026	27340
	3. AML (SM+WEM)	64,30%	67,00%	65,60%	6274	4035	6026	

Table 4.5: Results of produced alignments with **large FMA** as source and **SNOMED** as target ontology, the pre-trained embedding type and matcher. corresponds to String Matcher, WEM to Word Embeddings Matcher, and finally, SM+WEM corresponds to the combination of String and Word Embeddings Matcher.

Finally, in the last produced alignment, large FMA with SNOMED, the WEM performs better than the

SM, except in one case. In these cases, precision and F-measure improve with the embeddings matcher even though the recall decreases in all cases. On the other hand, with SM+WEM, the precision and F-measure decrease in most cases and, contrarily, the recall increases in most.

The results were evaluated to understand if the embedding matcher strategy was better than the string matcher. In this regard, we calculated the difference between the embedding matcher results minus the string matcher results. In addition, we assessed the Average and Median for all metrics.

The aggregated results of precision, recall and f1-measure are presented in the table 4.6. The results suggest that the embedding auto matcher does not improve the already implemented string matcher. Looking at the precision aggregated results, it seems that in some cases, the word embedding matcher improves the performance of AML. However, when looking at the median, we can see that it is below 0% for those same cases. This evidence implies that, for the precision average, there are differences with higher amplitude (making the average value positive). Nonetheless, the string matcher surpasses the embeddings matcher performance in most cases, leading to a median below 0%.

There is also a significant similarity between the results of Pubmed + PMC, Pubmed + PMC + Wikipedia and the English Wikipedia word embeddings. On the other hand, the BioWordVector Extrinsic seems to have performed better in this task than the BioWordVector Intrinsic since it has an average and f1-measure with less amplitude.

In terms of richness in words of the tested pre-trained word embeddings, for the ontologies used in this dissertation, BioWordVec Intrinsic and Extrinsic turn out to be the best files for these types of tasks. The aggregated results for the nulls count follow in the Table 4.7:

Pre-trained Word Embedding	**Total Nulls**
Intrinsic	37615
Extrinsic	37619
Pubmed + PMC	73248
Pubmed + PMC +Wikipedia	71041
English Wikipedia	88935

Table 4.7: Total count of words not found in each pre-trained word embedding file

From understanding how all entities and concepts are related inside an ontology to finding solutions to compare analogous entities from different ontologies, it seems clear that this represents a task that will always require a significantly complex solution.

When comparing English Wikipedia with the others pre-trained word embeddings, there seems to be no clear evidence that the size of the dimension of the vectors directly influences the results. Moreover, it performs as well as the word embeddings that are more related to the biomedical domain. A possible explanation for the fact that these embeddings do not perform better than English Wikipedia would be because they have lower coverage, leading to fewer mappings and, consequently, a higher precision.

Metric	Pre-trained Word Embedding	Average of Improvement of WEM over SM	Median of Averages of% WEM over SM
Precision	Intrinsic	-8,00%	-6,10%
	Extrinsic	3,78%	-0,80%
	Pubmed + PMC	1,72%	-0,20%
	Pubmed + PMC +Wikipedia	2,22%	0,00%
	English Wikipedia	1,58%	-0,40%
	Total	**0,26%**	**-0,40%**
Recall	Intrinsic	-1,44%	-1,50%
	Extrinsic	-1,98%	-1,20%
	Pubmed + PMC	-2,16%	-1,70%
	Pubmed + PMC +Wikipedia	-2,14%	-1,70%
	English Wikipedia	-2,18%	-1,80%
	Total	**-1,98%**	**-1,70%**
F1-measure	Intrinsic	-4,88%	-3,00%
	Extrinsic	0,26%	-1,30%
	Pubmed + PMC	-0,48%	-0,80%
	Pubmed + PMC +Wikipedia	-0,22%	-0,70%
	English Wikipedia	-0,54%	-1,10%
	Total	**-1,17%**	**-1,10%**

Table 4.6: Precision, Recall and F1-measure aggregated results according to the different pre-trained word vectors techniques. These calculations took into account the improvement of WEM by making the difference between the WEM and the SM results.

In some cases, the Word Embedding Matcher improved AML's results, like in the Mouse-Human, large FMA-NCI and large FMA-SNOMED tasks. For example, in the Mouse-Human task, there is one case where the F-measure increases when the word embedding is combined with the String Matcher (SM+WEM). As to the second task (large FMA-NCI), the WEM alone did improve the F-measure results in one case, and the SM+WEM improved another. Lastly, in the large FMA-SNOMED task, the WEM showed the best results by improving four of the five cases. For the same task, the SM+WEM improved the recall but a lower F-measure and precision.

As mentioned in section 2.2.1 it was expected that word embeddings could carry the semantic meaning of words more efficiently than a string similarity matcher. It was anticipated that when faced with cases where the string similarity was low but the mapping was correct (*e.g.* when ontologies use different terms for the same concept), the word embedding would be better at identifying it as a true mapping. However, the evaluation revealed the opposite. Word embeddings did not generally increase recall, even though they found more mappings than String Matcher. In most cases, using embeddings decreases precision, which can be explained by the fact that word embeddings capture relatedness and not necessarily equivalence, resulting in more mappings found but that are not actual equivalences [15].

However, as mentioned above, they do increase precision in the case of large FMA-SNOMED. This improvement can be explained by the fact that word embeddings attribute lower scores to ontology pairs entities with very similar names but whose names can be used with different meanings, thus producing vector representations with lower similarities. Since there is more focus on the meaning of the word and not on string similarity, the probability of being more accurate at finding true mappings increases in ontologies as big as large FMA-SNOMED [21]. This phenomenon is common in large ontologies with a smaller domain overlap, where the same term in one ontology has one meaning and in the other has another. For instance, in FMA, the class 59762 has one main label, 'Gingiva' and a corresponding synonym, 'Gum'. In SNOMED, the class 426210003 has the main label 'Gum'. While the String Matcher assigns a high similarity to these classes, the similarity is much reduced using the Word Embeddings Matcher.

Nevertheless, the overall results show that the string matcher remains slightly more efficient at finding relations between two concepts from two different ontologies than the word embeddings matcher. Additionally, the AML SM+WEM rarely performed better than when only using string or word embeddings matcher.

Although more sophisticated matching algorithms based on word embeddings can be employed, this study revealed that for the use cases that were evaluated, word embeddings do not bring substantial improvements over state-of-the-art ontology matching algorithms. Moreover, although there are improvements when using domain-specific pre-trained embeddings, these are also not marked. These results may reflect the fact that word embeddings which are trained to capture relatedness and contextual similarity [21] although successful in many Natural Language Processing tasks are not directly transferable to finding equivalent classes to perform ontology matching.

Chapter 5

Conclusions

This dissertation investigated the impact of using pre-trained word embeddings to measure the similarity between ontology concepts and find equivalence mappings. Word embeddings based similarity was integrated into AML, a state of the art ontology matching system. The expectation was that word embeddings trained on general and domain-specific *corpora* would improve the ontology matching performance, specifically by increasing recall and finding mappings between classes with different names but the same meaning.

Five different sources of pre-trained word embedding models were evaluated on the alignment of five ontology pairs. This evaluation showed that, in general, word embeddings did not increase recall, even though they found more mappings than the baseline approach. In most cases, the word embeddings caused a decrease in precision, resulting in more false mappings, which can be a result of word embeddings finding relatedness rather than equivalence. Word Embeddings Matching was more successful with large ontologies with a lower degree of overlap between them and with some dissimilar domains. In these cases, precision was increased, which can be explained by word embeddings giving a lower similarity to polysemous words. No substantial difference between using general or domain-specific embeddings was seen.

This study showed that the promise of pre-trained word embeddings for ontology matching is still unfulfilled, and that future work should focus on distinguishing between relatedness and equivalence when training embeddings for specific purposes.

Understanding how words should be correctly represented in a mathematical form is still challenging. Words are how people communicate with each other, and it is an abstract concept. Thus, it is still complex to understand and find the best way to put such an abstract idea into a language that can be processed and computed with the current technology. The N-dimensional space that best represents a concept is by itself already a complex task, for example. This problem becomes much more challenging, especially when dealing with complex data, such as biomedical data.

5.1 Future Work

As the next possible steps, a few things should be considered. AML usually runs with background knowledge sources. For this study, this option was deactivated to better support a comparison between standard ontology matching algorithms and the word embeddings. However, background knowledge sources can be used to enrich the vocabulary of the ontologies to match, which can in turn also help handle the missing embeddings for some words. Moreover, more sophisticated ways to combine WEM with AML could be explored. In this work, WEM was used instead of SM or in addition to it, without other changes to the AML standard pipeline. Furthermore, other ontology matching systems, less sophisticated than AML may stand to benefit more from word embeddings. Finally, it could also be interesting to explore the possibility of contextual embeddings, *e.g.* Bidirectional Encoder Representations from Transformers (BERT), for ontology matching in order to improve word embeddings models. It is possible that these methods will be better suited to handle the high prevalence of homonyms and synonyms in biomedical ontologies.

References

[1] S. Alshahrani, E. Ahmed, and R. Ward. The influence of online resources on student–lecturer relationship in higher education: a comparison study. *Journal of Computers in Education*, 4(2): 87–106, 2017. ISSN 2197-9987. doi: 10.1007/s40692-017-0083-8. 10

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001. URL http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21. 5

[3] K. W. Church. Emerging Trends: Word2Vec. *Natural Language Engineering*, 23(1):155–162, 2017. ISSN 14698110. doi: 10.1017/S1351324916000334. 10

[4] I. F. Cruz. AgreementMaker : Efficient Matching for Large Real-World. (2):1586–1589. 7

[5] I. F. Cruz, F. P. Antonelli, and C. Stroe. AgreementMaker: Efficient matching for large real-world schemas and ontologies. *Proceedings of the VLDB Endowment*, 2(2):1586–1589, 2009. ISSN 21508097. doi: 10.14778/1687553.1687598. 7

[6] M. Diller, E. Johnson, A. Hicks, and W. R. Hogan. A realism-based approach to an ontological representation of symbiotic interactions. *BMC Medical Informatics and Decision Making*, 20(1): 1–16, 2020. ISSN 14726947. doi: 10.1186/s12911-020-01273-0. 1

[7] J. Euzenat and P. Shvaiko. *Ontology matching, 2nd Edition*. 2013. ISBN 3540496114. URL http://book.ontologymatching.org/. 6, 7

[8] W. Fang, L. Ma, P. E. Love, H. Luo, L. Ding, and A. Zhou. Knowledge graph for identifying hazards on construction sites: Integrating computer vision with ontology. *Automation in Construction*, 119(May):103310, 2020. ISSN 09265805. doi: 10.1016/j.autcon.2020.103310. URL https://doi.org/10.1016/j.autcon.2020.103310. 6

[9] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, and F. M. Couto. Matching System. pages 527–541, 2013. 9

[10] D. Faria, C. Pesquita, E. Santos, I. F. Cruz, and F. M. Couto. AgreementMakerLight: A Scalable Automated Ontology Matching System. *Proceedings from the 10th International Conference on Data Integration in the Life Sciences (DILS 2014)*, (July):1–4, 2014. XI, 9

[11] D. Faria, C. Pesquita, I. Mott, C. Martins, F. M. Couto, and I. F. Cruz. Tackling the challenges of matching biomedical ontologies. *Journal of Biomedical Semantics*, 9(1):1–19, 2018. ISSN 20411480. doi: 10.1186/s13326-017-0170-9. 2, 7, 9

[12] D. Faria, C. Pesquita, I. Mott, C. Martins, F. M. Couto, and I. F. Cruz. Tackling the challenges of matching biomedical ontologies. *Journal of biomedical semantics*, 9(1):1–19, 2018. 2

[13] T. Iesmantas, A. Paulauskaite-Taraseviciene, and K. Sutiene. Enhancing multi-tissue and multi-scale cell nuclei segmentation with deep metric learning. *Applied Sciences (Switzerland)*, 10(2), 2020. ISSN 20763417. doi: 10.3390/app10020615. 10

[14] Z. Imtiaz, M. Umer, M. Ahmad, S. Ullah, G. S. Choi, and A. Mehmood. Duplicate Questions Pair Detection Using Siamese MaLSTM. *IEEE Access*, 8:21932–21942, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.2969041. 10

[15] D. Kiela, F. Hill, and S. Clark. Specializing word embeddings for similarity or relatedness. *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, (September):2044–2048, 2015. doi: 10.18653/v1/d15-1242. 9, 26

[16] P. Kolyvakis, A. Kalousis, B. Smith, and D. Kiritsis. Biomedical ontology alignment: An approach based on representation learning. *Journal of Biomedical Semantics*, 9(1):1–20, 2018. ISSN 20411480. doi: 10.1186/s13326-018-0187-8. 11

[17] O. Lassila and R. R. Swick. Resource description framework (RDF) model and syntax specification. World Wide Web Consortium Recommendation. (October), 1999. URL `http://www.w3.org/TR/REC-rdf-syntax`. 5

[18] B. Li, A. Drozd, Y. Guo, T. Liu, S. Matsuoka, and X. Du. Scaling Word2Vec on Big Corpus. *Data Science and Engineering*, 4(2):157–175, 2019. ISSN 23641541. doi: 10.1007/s41019-019-0096-6. URL `https://doi.org/10.1007/s41019-019-0096-6`. 10

[19] G. Li. Improving Biomedical Ontology Matching Using Domain-specific Word Embeddings. *PervasiveHealth: Pervasive Computing Technologies for Healthcare*, 2020. ISSN 21531633. doi: 10.1145/3424978.3425102. 2, 10, 12

[20] B. Lima, D. Faria, F. M. Couto, I. F. Cruz, and C. Pesquita. OAEI 2020 results for AML and AMLC. *CEUR Workshop Proceedings*, 2788:154–160, 2020. ISSN 16130073. 9

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pages 1–12, 2013. XI, 10, 11, 26

[22] X. Rong. word2vec Parameter Learning Explained. pages 1–21, 2014. URL http://arxiv.org/abs/1411.2738. 10

[23] E. Santos, D. Faria, C. Pesquita, and F. M. Couto. Ontology alignment repair through modularization and confidence-based heuristics. *PLoS ONE*, 10(12):1–19, 2015. ISSN 19326203. doi: 10.1371/journal.pone.0144807. 8

[24] F. Z. Smaili, X. Gao, and R. Hoehndorf. OPA2Vec: Combining formal and informal content of biomedical ontologies to improve similarity-based prediction. *Bioinformatics*, 35(12):2133–2140, 2019. ISSN 14602059. doi: 10.1093/bioinformatics/bty933. 2, 10

[25] S. Staab and R. Studer. *Handbook on ontologies*. Springer, Berlin, 2009. ISBN 9783540926733 3540926739. URL http://public.eblib.com/choice/publicfullrecord.aspx?p=571805. 5

[26] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge Engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197, 1998. ISSN 0169023X. doi: 10.1016/S0169-023X(97)00056-6. 5

[27] D. Szklarczyk, J. H. Morris, H. Cook, M. Kuhn, S. Wyder, M. Simonovic, A. Santos, N. T. Doncheva, A. Roth, P. Bork, L. J. Jensen, and C. Von Mering. The STRING database in 2017: Quality-controlled protein-protein association networks, made broadly accessible. *Nucleic Acids Research*, 45(D1):D362–D368, 2017. ISSN 13624962. doi: 10.1093/nar/gkw937. 1

[28] y. zhang, q. chen, z. yang, h. lin, and Z. Lu. Biowordvec: Improving biomedical word embeddings with subword information and mesh ontology, Sep 2018. URL https://figshare.com/articles/dataset/Improving_Biomedical_Word_Embeddings_with_Subword_Information_and_MeSH_Ontology/6882647/2. 16