Comparing Word Embeddings through Visualisation

Pedro Santos*, Nuno Datia*†, Matilde Pato*‡, José Sobral*
*ISEL - Lisbon School of Engineering
Politécnico de Lisboa
Lisbon, Portugal
†NOVA LINCS, NOVA School of Science and Technology
Monte da Caparica, Portugal
‡LASIGE, FCUL, Universidade de Lisboa
Lisbon, Portugal
Email: *pedro.santos@deetc.isel.ipl.pt, *†datia@isel.ipl.pt, *‡matilde.pato@isel.pt, *jose.sobral@isel.pt

2022 26th International Conference Information Visualisation (IV) | 978-1-6654-9007-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/IV56949.2022.00024

***Abstract*—Asset management is a branch of facilities management that is responsible for the operation and maintenance of assets. The most common means of managing assets and their life-cycle is through requests and work orders. A request is used to report an occurrence that is detected either by a sensory device, a technician, or non-technical personnel; they are used to pointing out that something is wrong in a given asset, and needs appropriate attention. Depending on the problem, a request can give rise to a work order if the solution is not trivial. Work orders consist in technical reports that specify the asset that needs intervention and has the details about the work to be done or, in the case that the work is unknown from the start, the characteristics of the malfunctioning. Work orders contain a set of words, free text, that are not restricted from a fixed set of vocabulary, making it difficult to automatically analyse them. In this paper, we discuss the application of modern Natural Language Processing techniques to process the work order's description, while presenting a comparison between two *Word Embedding* models — *Word2Vec* and *Fasttext*— through semantic similarity tests between the encoded words, and a visualisation of the vector space through dimensionality reduction of the encoded vectors. The results show a better performance of the *Fasttext* approach, considering the semantics of the results.**

***Index Terms*—NLP, Word Embeddings, Visualisation, Asset Management**

I. Introduction

For every occurrence on an asset, either urgent or not, there must be an request, followed by the creation of a work order. Thus, the creation and management of requests and work orders allows one to achieve a regulation of the asset's life cycle. Human-generated data may have problems, such as quality issues or inconsistencies, which can be problematic. When creating a work order, the sentences created are considered to be "free text", left to the criteria of the technician who wrote them. It can lead to spelling errors, non-existent words, or inconsistent usage of terms, i.e., the use of different terms to describe the same job in work orders that are the same. These are some of the problems that arise when dealing with data related to natural language, so strategies must be developed to overcome them.

The first solution that might come to mind is the usage of a rigid taxonomy, so words inputted by the technicians are no longer unchecked. This solution fails because it turns the creation of a work order into a time-consuming process. As stated above, having the sentences of each work order as free text makes it harder to discover the tendencies and carry out the analysis process. In light of this, it is important to standardize the terms and words used by technicians. Therefore, subjective sentences can be excluded, and the totality of the actions performed in assets can be fully covered. The standardization can be achieved by suggesting words to a technician as he fills out a work order description. The words that are suggested are dependent on the asset, on already existing work order words. Each time a new word is entered, the suggestion algorithm recalculates the words that are shown, taking into account the context in which the work order is created (e.g. the targeted asset, the maintenance area). To achieve this goal, dedicated research is carried out, mainly in the Natural Language Processing [1] domain, in which a comparison is made between two different models of word embeddings [2].

The objective of this paper is to compare two models of word embeddings, *Word2Vec* [3] and *Fasttext* [4], through the visualisation of each vector space and through the querying of selected words. This querying has the goal of evaluating how well the models behave in terms of measuring word similarity. The contributions are the application of the studied *NLP* techniques to the asset management domain and the discovery of a visual pattern, formed by the words in the work orders of a specific domain — asset management in a Portuguese health facility.

This article is organised as follows: Section two presents the research background prior to this research, Section three describe the data used, Section four presents the modelling of the problem and the steps taken towards the objective. The last section concludes the work and points out future work directions.

This work was supported by the REV@CONSTRUCTION mobiliser project, under the grant LISBOA-01-0247-FEDER-046123 from ANI - National Innovation Agency, and by NOVA LINCS (UIDB / 04516/2020) and LASIGE (UIDB / 00408/2020) with financial support from FCT— Fundação para a Ciência e a Tecnologia, through national funds.

2375-0138/22/$31.00 ©2022 European Union
DOI 10.1109/IV56949.2022.00024

II. Research Background

The concept of word embeddings is not new in the landscape of natural language processing. In recent years, several word embedding models have been developed, such as *Word2Vec* [3], *Fasttext* [4], *GloVe* [5], among others [6]. The difficulty lies in understanding which word embeddings model gives a better fit for the given problem at hand, since each model has its own intricacies and each problem has its own complexity. Furthermore, to better decide which word embedding model to use in each situation, it is important the performance and results of these models are compared in different scenarios. Schnabel et al. [7] presents a comprehensive study of evaluation methods for unsupervised embedding techniques, pointing out that there is not always a single, more efficient, go-to solution. As a result of different evaluations, the embedding methods are ordered in different ways. Wang et al. [8] empirically evaluates word embeddings trained from four different corpora, namely clinical notes, biomedical publications, *Wikipedia*, and news. For the case of clinical notes and biomedical publications, word embeddings were trained using unstructured electronic health record (EHR) data available at *Mayo Clinic*, and articles (*MedLit*) from *PubMed Central* [9], respectively. For *Wikipedia* and the news, pretrained word embeddings *GloVe* [5] and *Google News* were used. The conclusions drawn were that the word embeddings trained in *EHR* and *MedLit* have the ability to better capture the semantics of medical terms and to find semantically relevant medical terms closer to human judgements than those trained in *GloVe* and *Google News*. It was also concluded that there is no consistent global ranking of word embeddings for all downstream biomedical *NLP* applications. As a last conclusion, word embeddings trained in the biomedical domain corpora do not necessarily have better performance than those trained in the general domain corpora, for any given downstream biomedical Natural Language Processing task. Bhoir et al. [10] presents a comparative analysis of different word embedding models: *Continuous bag of words* [3], *Skip gram* [3], *Glove(Global Vectors for word representation)* [5] and *Hellinger — PCA (Principal Component Analysis)* [11]. The models are compared, having into account different parameters, which include performance with respect to the size of training data, basic overview, the relation of context and target words, memory consumption, the supported classifier used, and effect of changes in dimensions. After an extensive comparison of each of the models taking into account the mentioned metrics and after weighting all the pros and cons, authors conclude that *GloVe* [5] is the best model compared to other models. The decision is supported on how the model scales to large corpus, but it also works well with small corpus; it improves the quality of learned representations by normalising counts and log smoothing them.

Several studies have shown that machine learning can improve the asset management process, but they did not find a way to implement a specialised word suggestion algorithm based on the data collected from these activities. This article aims to explore the potential of machine learning to improve the way work orders are described in a language that is commonly used, exploring a visualisation in a vector space of the words used, for a specific domain — a Portuguese health facility.

TABLE I: Work Order's important fields and examples

Field Name	**Sample Example**
Word Order ID	16715
Job Number (*work orders may have may jobs*)	1
Asset Complete Identifier	"06001MCTEEE"
Asset ID	6001
Work Order Description	"Periodic review of leaks"
Date when the work order began	"2020-06-04T09:00:00"
Date when the work order ended	"2020-06-04T11:00:00"
Description of the work performed	"Periodic review of leaks"
Asset designation	"Chiller"
Asset Family	"CT"
Asset Subfamily	"CH"

III. Data Description

Data used was extracted from maintenance records collected from a health facility in Portugal. Each maintenance record is used to characterise and keep history of a given situation in the context of the healthcare facility (e. g. asset malfunction, routine inspection, component substitution). Therefore, it can be used to analyse tendencies and relationships between the words that are used to describe the situation/problem. These maintenance records consist of work orders which are composed of several fields and in which the technician fills in the information to best describe the given situation. The data set consists of a total of 38.445 work orders, all related to the health facility and its assets. Of all the seventy five fields that compose a work order, the ones with the most significant information are presented hereafter, with an example for each field, taken from a real work order (see TABLE I). From the information present in these fields, it is possible to perform an analysis of the lexicon used by the technicians and to study the relationships between words used to describe day-to-day maintenance situations in the Portuguese health facility. All work orders present in the healthcare facility's *Computerized Maintenance Management System* (CMMS) were gathered into a single *JSON* file, from which it was possible to read and parse the work orders and their most significant fields into memory. No preprocessing of the work orders was done, except selecting the most relevant fields. There are no standard terms in the work order descriptions written by technicians, so each describes a given situation in his own way, resulting in a variety of possible descriptions of the same problem. Despite the differences in words, the asset management situation is the same. The objective is to create a consistency in the language used when the same maintenance situation is addressed. By suggesting words as a technician fills in a description, consistency is maintained. This suggestion is based on the words already entered and knowledge of maintenance situations, their descriptions, and the words used.

In order to accomplish this, a *NLP* pipeline is created that uses work orders as input. The core objective of this pipeline

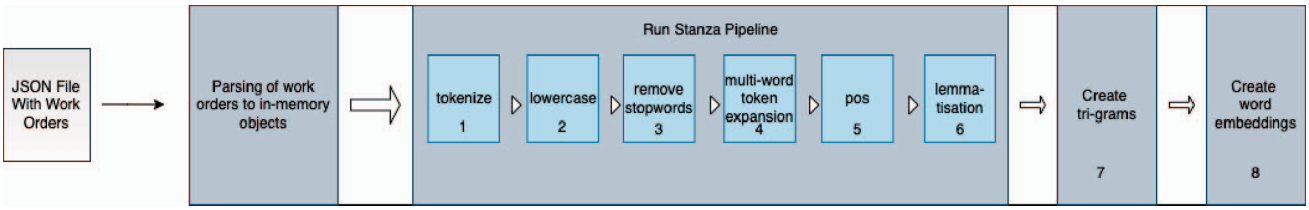


Fig. 1: *NLP* Pipeline created

is to gather linguistic information about the words used in the descriptions and the relationships established between them. In the next section, we will describe it step-by-step. For this study, only the *Work Order Description* attribute was used.

IV. Modelling

The parsing of the work orders into memory allows them to be entered into the *NLP* pipeline illustrated in Fig. 1, which has the objective of analysing the fields present in each work order and enable us to draw conclusions about the words used and the relationships between them. The pipeline is created, in part, using the Python Stanza framework [12].

A. Natural Language Processing Pipeline

The next sections will describe the steps of the *NLP* pipeline described in Fig. 1.

1) Tokenization: The first applied technique is *tokenization* [13], which is a way to separate a piece of text into smaller units called tokens. In this particular case, *tokenization* is achieved by splitting words by spaces and keeping the resulting tokens. Multi word tokens such as *life-cycle* were dealt separately, in another phase.

2) Lowercase: Module designed to convert every description to lowercase letters.

3) Stopword Removal: After applying lowercase, it is performed the removal of words that do not have relevance in the problem domain (commonly known as *stopwords* [14]). The main motive behind the elimination of *stopwords* is to increase the execution speed and the accuracy of the subsequent algorithms, since only relevant words are considered. Here, we have a domain expert indicating which words should be discarded.

4) Multi-word token expansion: The multi-word token expansion module is used to treat compound tokens that are not processed during the tokenisation module. These are tokens such as *life-cycle*; in this case, *life-cycle* is considered as a token composed by words *life* and *cycle*.

5) POS Tagging: Part-of-speech (*POS*) tagging [15] enables to produce important indicators that will enable to understand the relationships between words, what each word represents in each sentence, and how each word influences the overall meaning of a sentence. It works by categorising words in a text (corpus) in correspondence with a particular part of speech, depending on the definition of the word and its context.

6) Lematisation: From the knowledge gathered by *POS Tagging*, it is then performed *lemmatisation*, a process of grouping together the inflected forms of a word, identified as the *lemma*. For example, the word *chillers* has as lemma *chiller*, because the process of lemmatisation converts the words from plural to singular form. Another good example are verbs, in which conjugated verbs are converted to its infinitive form (e.g *grouping* is converted to *group*, *fixed* to *fix*). This process is commonly used to minimise the number of possible words that the subsequent algorithms deal with. This reduces significantly the complexity and variance present in the data.

7) Creation of n-grams: A *n-gram* is a combination of n words that often occur together. The purpose is to maintain together words that are often together. In this particular module, bi-grams and tri-grams are generated. The algorithm specified to find n-grams concatenates these words using an underscore character. For example, the description *preventive maintenance report in annex* is transformed into *preventive maintenance report annex* after the stopword removal module. Then, in the n-gram step, all its words are concatenated using underscores, due to the fact that the four words often appear together. So, the final result is *preventive_maintenance_report_annex*.

8) Word embeddings: Computers operate mostly using numeric representations, so they do not understand textual representations the way humans do. A way to overcome this problem is to convert each word (including the n-grams) into a numeric vector that tries to capture the meaning of the word through numbers, taking into account its context. Thus, for words with similar meaning we have a similar numerical representation.

B. Creation of word embeddings

There are several libraries that can be used to generate them from a corpus of text. The ones chosen were *Word2Vec* [3] and *Fasttext* [4], which were used to create word embeddings from scratch. There was no pretension of using pre-trained word embeddings due to the fact that the lexicon found in the domain problem is technical and too specific [16]. Both the *Word2Vec* and *Fasttext* models were generated using the descriptions resulting from the *n-gram* model.

Word2Vec is a two-layer neural network that is trained to reconstruct linguistic contexts of words. It does so by taking a corpus of words and producing a vector space with a large number of dimensions (usually between 100 and 300 dimensions are recommended, depending on the complexity of the corpus). For each unique word in the corpus, a corresponding

Fig. 2: Plot of the Word embeddings learned by the *Fasttext* model, in 2D

Fig. 3: Plot of the Word embeddings learned by the *Word2Vec* model, in 2D

vector in the space is assigned. Word vectors are positioned in the vector space such that words sharing similar meaning in the corpus are closely located in the vector space. It is possible to generate the *Word2Vec* model from two different architectures [3]: 1) *Continuous Bag of Words (CBOW)*, and 2) *Skipgram*. CBOW predicts target words from surrounding context words, while Skipgram predicts surrounding context words from target words (inverse of CBOW). There is a single hidden layer in the *Word2Vec* neural network architecture. Its goal during training is to adjust its weights to reduce a loss function. However, the outputs of the neural network are not considered, and, instead, the hidden weights are used as the word embeddings.

Fasttext uses the same principles of *Word2Vec* but with a slight difference: it operates at the character level, while *Word2Vec* operates at the word level. Therefore, the vector for a word is made up of the sum of the word's constituents. For example, the word vector *matter* is a sum of the vectors of the constituents <*ma*, *mat*, *att*, *tte*, *ter*, *r*>, where '<' means *Start of Word* and '>' *End of Word*.

Both libraries are designed as a neural network and provide the *CBOW* and *Skipgram* architectures. However, *Fasttext* uses enrichment of word embeddings using sub-word information [4].

C. Visualisation of the word embeddings

Both libraries are put to the test and word embeddings are generated using the work order's descriptions. Both the *Fasttext* and *Word2Vec* models are generated using standard values for the parameters, and from the total of 38.445 descriptions, 2435 words are learned in both models. Since each vector (word embedding) has 100 dimensions, to properly visualise the vector space it is necessary to conduct a dimensional reduction. The techniques used to reduce the dimensions of the vector space in 3D and visualise the position of each word embedding were Principal Component Analysis (PCA) [17] first, followed by t-distributed stochastic neighbour embedding (t-sne) [18]. The reason for using *PCA* before applying *t-sne* is that it allows to suppress some noise and speed up the computation of pairwise distances between samples. It is important to notice that the t-sne dimensionality reduction technique operates in a stochastic manner, so for each creation of a plot, there are slightly different representations. This happens due to the fact that one of the first steps of the t-sne algorithm is to select a random point in space, so the algorithm reduces the dimensions having into account that first selected point. After computing several plots, it is possible to observe a common trend in all of them: words in the 3D space are displayed following a helical shape, both in the *Fasttext* model and the *Word2Vec* model. The 3D representations are dense, and difficult to analysis without an interactive interface. Thus, for displaying purposes, a dimensional reduction for two dimensions was generated, as depicted in Fig.s 2 and 3. Nevertheless, the same conclusions can be taken.

The first impression that comes to mind when observing both images is that the words are displayed in a helical shape, but with slight differences between the two models. In the case of the *Fasttext* model the resulting spiral is precisely drawn, especially in the lower left corner, where the words are placed so that they almost form a straight line. In the plot related

to the *Word2Vec* model, a similar spiral is observed. For the *Word2Vec* case, the spiral is drawn in a less precise way, that is, the words are not placed in a way that resembles a straight line. Accordingly, the representations appear to be mirror images of each other because the dimensional reduction technique t-SNE [18], which was applied, is stochastic in nature. The reason for a less precise placement of the words in the case of the *Word2Vec* model might be explained by a lower accuracy of the *Word2Vec* algorithm in determining with precision the meaning of each word, that is, the computed word embeddings aren't as precise as the *Fasttext* ones. This probably occurs due to the fact that the latter uses sub-word information to generate the final word embeddings, while *Word2Vec* only uses the words. This hypothesis is analysed and discussed further in this document.

Moreover, it is important to analyse the meaning of a helical shape in the context of the visualisation of a word embeddings model. The fact that the observed shape is a spiral helps to understand that the corpus of words where the model was trained on do not present any groups/topics. If that was the case, there would be dense groups of words (e.g. a circular shape made of several words), which meant that the words present in that group define an abstract concept in the context of the domain (e.g. if the domain was food then a topic that represented the concept of *fruit* would have the words *apple*, *peach*, *pear* placed together, in a dense way). Since the shape is a spiral, defined by semi-straight lines, it is possible to conclude that there are no topics in the descriptions of the word orders. The words are dispersed across different areas of expertise (e.g. mechanics or electricity), and the same word is used to describe different situations, so no groups are identified.

D. Visualisation of Tag Clouds

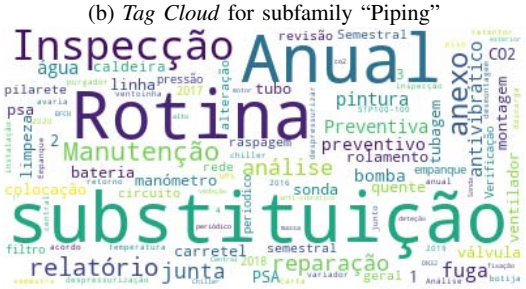
In the context of the case study, there are assets that cover a total of eighty different subfamilies[1]. The 38.445 work orders are filtered and grouped by subfamily, and each description is used to generate a *tag cloud* per asset subfamily (from Fig.s .4a to 4c) *Tag clouds* consist in a visual representation of words, where each word appears represented according to its frequency in the corpus. The work orders descriptions are used to create the tag clouds, and it is possible to observe the most frequent words standing out due to the size of the font and/or the colour. A tag cloud is also computed using all the work orders (Fig. 4a), despite their subfamily, to have a general view of the lexicon used. The words are in Portuguese because the main objective of presenting these tag clouds is to analyse the frequency of each word in the subfamilies *Piping* (Fig. 4b), *Chiller* (Fig. 4c) and *Gas Centrals* (Fig. 4d), along with the tag cloud that provides a general view of the lexicon (Fig. 4a).

The words that stand out instantly in all of the tag clouds, thus for any arrangement of work orders, are the same, namely, *substituição*, which means **substitution**, *inspeção*

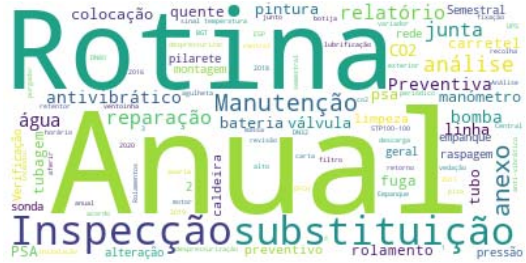
[1]A subfamily is a domain specific 2º level classification

(a) *Tag Cloud* computed for all the work orders

(b) *Tag Cloud* for subfamily "Piping"



(c) *Tag Cloud* for subfamily "Chiller"



(d) *Tag Cloud* for subfamily "Gas Centrals"

Fig. 4: Illustration of some of the generated *Tag Clouds*

which means **inspection**, *reparação* which means **repair** and others less noticeable like *rotina*, meaning **routine**, and *anual*, meaning **annual**. It is curious that in three very different asset subfamilies, the more frequent words are practically the same, just as in the global tag cloud (Fig. 4a). A possible cause for this finding is that the listed words can be used generically in the domain. For example, the word *substitution* can be used in a multitude of situations where it is necessary to perform the replacement of some component. This leads to the dissemination of the word across all subfamilies. This consideration also allows us to corroborate the finding discussed in the previous subsection, regarding the word embeddings

models visualisation: the words present in the corpus are used dispersedly, and the same words are used in different contexts, meaning there are no groups of words used in specific contexts. The same words are used to describe very different asset maintenance situations, making them less informative. So the differences between the lexicon used in these three subfamilies can only be detected when observing the less frequent words.

From the observation of the tag clouds, it is also possible to conclude that a general machine learning model trained on the entirety work orders descriptions will not likely yield good results, due to the fact that the global tag cloud presents roughly the same most important words as the three subfamilies, so it may have difficulty in identifying subfamily-specific data that is required for it to learn properly. Therefore, it is necessary to consider other solutions, such as a machine learning model per subfamily. The downside of this solution is that it requires managing a large number of models, but it has the advantage of retaining the specific details that are unique to each subfamily.

Another important consideration is that the frequency with which words appear is not the only indicator of importance. Domain knowledge allows one to understand that words that rarely appear (e.g. one or two times) might be equally as important as words that appear very often. This happens due to the fact that critical failures are not common in asset management, so words used specifically in these situations won't appear very often. However, when these critical failures occur, they present very important information that needs to be included in asset management decision-making processes.

E. Performance of the word embeddings models

The basis for this research is to determine how to compare the performance of the word embeddings generated by each library, while having into account the domain where these techniques are applied and what conclusions can be drawn from their application.

The comparison of both word embeddings models is made through the querying of words that are semantically similar to a given word, assessing each model's response, and studying whether the returned words are coherent. This is done after the word embedding models are created, and using specific methods in each library that allow to find the closest words (in the vector space) to a given inputted word. While the semantics of the word and the domain problem cannot be objectively compared, they do need to be qualitatively compared. The first example of the words searched is the word *2020*. On the one hand, the *Fasttext* model returned the words *2019*, *2017*, *2018* and *2016* as the closest words. On the other hand, the *Word2Vec* model returned the words *alarm*, *cable_passage*, *anomaly*, and *collage*. The second example is *break_room*, and the words returned by *Fasttext* were *delivery_room*, *work_room*, *waiting_room* and *room*. The words returned by *Word2Vec* were *floor_six*, *dressing_room*, *fall* (verb) and *fix*. The third example is *employee*, and the words returned by *Fasttext* are *schedule*, *malfunction*, *employee_entrance* and *necessary*. The words returned by *Word2Vec* are *sug* (not a word, probably it is an acronym from the domain), *badly*, *together* and *key*. The last example is a more curious case to analyse due to the fact that the words returned by the *Fasttext* model were not as strongly semantically linked, but it is possible to understand that there is a relationship between an *employee* (queried word) and the returned words — *schedule*, *malfunction*, *employee_entrance* and *necessary*. For the cases of the words *malfunction* and *employee_entrance* it is possible to deliberate that the *Fasttext* algorithm captured the existing relationship between the queried word and these responses, since a malfunction is fixed by an employee and the entrance of an employee has the concept of employee in itself. For the other two cases, it is not possible to assume whether the *Fasttext* model captured the relationship between an employee and a schedule (an employee has a schedule) and between an employee and the concept of necessity (an employee is needed to perform some operation), or if there was a matching of sub-word information, since in Portuguese the three words (employee, schedule and necessary) end with the same four letters, that is, funcionário, horário and necessário, respectively.

The exploration of many examples, where these three are just a representative sample, it was possible to conclude that semantically, the accuracy of the word embeddings generated by the *Word2Vec* model was poor. This is visible because the returned words had little or nothing to do with the queried word in most of the examples. On the contrary, the *Fasttext* model responded to the queried examples with words that generally made sense — are semantically similar — or are related. Either way, the results provided by the *Fasttext*'s word embeddings made sense in most of the queried examples, although in more technical words from the domain the returned words did not make much sense, so there is room for improvement that can be further explored by tweaking the parameters of the *Fasttext*'s train algorithm.

V. Conclusions

The utilisation of word embeddings was documented in this article, with a comparison on two different models, *Word2Vec* and *Fasttext*The conclusion was that with the given corpus of words the performance of *Fasttext* was superior. This was noticeable both in visual representations, where the helical shape was drawn more thoroughly, and in semantic similarity tests, where specific words were queried to the models. *Fasttext* did its job by responding words that made sense, while *Word2Vec* often return unrelated words. Although the performance of *Fasttext* was better than *Word2Vec* it still has some shortcomings when it comes to the more technical words. These shortcomings can be tackled with an empirical study of what values for the parameters of the model to use.

The fact that it is possible to visualise the word embeddings in the corresponding vector space enables us to draw conclusions that otherwise would need further computational explorations, such as the conclusion that there are no groups/topics in the corpus. This conclusion means that further exploration of NLP concepts is necessary, such as more complex Neural Networks, oriented to text generation, in order to properly

reach the ultimate objective of suggesting words while a technician writes a work order.

The use of *tag clouds* allows us to have a broad perspective of how the lexicon is spread across the several asset subfamilies; the conclusion is that words are used dispersedly throughout the asset subfamilies, so it is difficult to find a direct relationship between a set of words and an asset subfamily. In other words, it is not possible to identify what kind of asset is being treated only by looking at the words used in a given description, since the same words are used in a variety of contexts.

REFERENCES

[1] G. G. Chowdhury, "Natural language processing," *Annual Review of Information Science and Technology*, vol. 37, no. 1, p. 51–89, 2005.

[2] C. Allen and T. Hospedales, "Analogies explained: Towards understanding word embeddings," in *International Conference on Machine Learning*, pp. 223–231, PMLR, 2019.

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

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

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[6] Y. Li and T. Yang, "Word embedding for understanding natural language: a survey," in *Guide to big data applications*, pp. 83–104, Springer, 2018.

[7] T. Schnabel, I. Labutov, D. Mimno, and T. Joachims, "Evaluation methods for unsupervised word embeddings," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 298–307, 2015.

[8] Y. Wang, S. Liu, N. Afzal, M. Rastegar-Mojarad, L. Wang, F. Shen, P. Kingsbury, and H. Liu, "A comparison of word embeddings for the biomedical natural language processing," *Journal of biomedical informatics*, vol. 87, pp. 12–20, 2018.

[9] "Pub med central." https://www.ncbi.nlm.nih.gov/pmc/.

[10] S. Bhoir, T. Ghorpade, and V. Mane, "Comparative analysis of different word embedding models," in *2017 International conference on advances in computing, communication and Control (ICAC3)*, pp. 1–4, IEEE, 2017.

[11] R. Lebret and R. Collobert, "Word emdeddings through hellinger pca," *arXiv preprint arXiv:1312.5542*, 2013.

[12] "Stanza framework." https://stanfordnlp.github.io/stanza/.

[13] "What is tokenization: Tokenization in nlp," Jul 2021. [Online; accessed 9-December-2021].

[14] J. Kaur and P. K. Buttar, "A systematic review on stopword removal algorithms," *International Journal on Future Revolution in Computer Science and Communication Engineering*, vol. 4, no. 4, 2018.

[15] Wikipedia contributors, "Part-of-speech tagging — Wikipedia, the free encyclopedia," 2021. [Online; accessed 9-December-2021].

[16] P. Bojanowski, O. Celebi, T. Mikolov, E. Grave, and A. Joulin, "Updating pre-trained word vectors and text classifiers using monolingual alignment," *arXiv preprint arXiv:1910.06241*, 2019.

[17] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.

[18] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.