DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2020

Decentralizing Large-Scale Natural Language Processing With Federated Learning

DANIEL GARCIA BERNAL

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Decentralizing Large-Scale Natural Language Processing With Federated Learning

DANIEL GARCÍA BERNAL

Master's Programme, ICT Innovation, 120 credits
Date: July 1, 2020
Supervisors: Lodovico Giaretta (KTH), Magnus Shalgren (RISE Gavagai)
Examiner: Šarūnas Girdzijauskas
School of Electrical Engineering and Computer Science
Host company: RISE Gavagai
Swedish title: Decentralisering av storskalig naturlig språkbearbetning med förenat lärande

Decentralizing Large-Scale Natural Language Processing With Federated Learning / Decentralisering av storskalig naturlig språkbearbetning med förenat lärande

Abstract

Natural Language Processing (NLP) is one of the most popular and visible forms of Artificial Intelligence in recent years. This is partly because it has to do with a common characteristic of human beings: language. NLP applications allow to create new services in the industrial sector in order to offer new solutions and provide significant productivity gains. All of this has happened thanks to the rapid progression of Deep Learning models. Large scale contextual representation models, such as Word2Vec, ELMo and BERT, have significantly advanced NLP in recently years. With these latest NLP models, it is possible to understand the semantics of text to a degree never seen before. However, they require large amounts of text data to process to achieve high-quality results. This data can be gathered from different sources, but one of the main collection points are devices such as smartphones, smart appliances and smart sensors. Lamentably, joining and accessing all this data from multiple sources is extremely challenging due to privacy and regulatory reasons. New protocols and techniques have been developed to solve this limitation by training models in a massively distributed manner taking advantage of the powerful characteristic of the devices that generates the data. Particularly, this research aims to test the viability of training NLP models, in specific Word2Vec, with a massively distributed protocol like Federated Learning. The results show that Federated Word2Vec works as good as Word2Vec is most of the scenarios, even surpassing it in some semantics benchmark tasks. It is a novel area of research, where few studies have been conducted, with a large knowledge gap to fill in future researches.

Keywords

Natural Language Processing, distributed systems, Federated Learning, Word2Vec

Sammanfattning

Naturligt språkbehandling är en av de mest populära och synliga formerna av artificiell intelligens under de senaste åren. Det beror delvis på att det har att göra med en gemensam egenskap hos människor: språk. Naturligt språkbehandling applikationer gör det möjligt att skapa nya tjänster inom industrisektorn för att erbjuda nya lösningar och ge betydande produktivitetsvinster. Allt detta har hänt tack vare den snabba utvecklingen av modeller för djup inlärning. Modeller i storskalig sammanhang, som Word2Vec, ELMo och BERT har väsentligt avancerat naturligt språkbehandling på senare tid år. Med dessa senaste naturliga språkbearbetningsmo modeller är det möjligt att förstå textens semantik i en grad som aldrig sett förut. De kräver dock stora mängder textdata för att bearbeta för att uppnå högkvalitativa resultat. Denna information kan samlas in från olika källor, men en av de viktigaste insamlingsställena är enheter som smartphones, smarta apparater och smarta sensorer. Beklagligtvis är det extremt utmanande att gå med och komma åt alla dessa uppgifter från flera källor på grund av integritetsskäl och regleringsskäl. Nya protokoll och tekniker har utvecklats för att lösa denna begränsning genom att träna modeller på ett massivt distribuerat sätt med fördel av de kraftfulla egenskaperna hos enheterna som genererar data. Särskilt syftar denna forskning till att testa livskraften för att utbilda naturligt språkbehandling modeller, i specifika Word2Vec, med ett massivt distribuerat protokoll som Förenat Lärande. Resultaten visar att det Förenade Word2Vec fungerar lika bra som Word2Vec är de flesta av scenarierna, till och med överträffar det i vissa semantiska riktmärken. Det är ett nytt forskningsområde, där få studier har genomförts, med ett stort kunskapsgap för att fylla i framtida forskningar.

Nyckelord

Naturligt språkbehandling, distribuerade system, federerat lärande, Word2Vec

Contents

Chapter 1

Introduction

1.1 Background

It was back in 1969 when the architecture of the first Neural Network (NN) was designed: The Perceptron [1]. But it was not until the last years that these models started to be used, becoming more popular. This delay was caused by two main problems that the field of Machine Learning faced during the last century: the lack of data and the lack of computational capabilities. Both issues have started to disappear during the first half of this decade, leading to important improvements in models and algorithms in Machine Learning, and in the subfield of Deep Learning [2].

The growth in the amount of data and the way it can be used has played an important role in the development of Deep Learning. The data generated during this decade has continued to increase. Every day, new large amounts of data are created by users of smartphones, social media or searches on the internet. And not only by users, but also by sensors and machines thanks to the advances in the Internet of Things (IoT) sector. The same devices that generate the data also offer computational capabilities that increase every year.

1.1.1 Distributed vs data-private massively-distributed approaches

Datacenter-scale ML algorithms work with vast amounts of data allowing the training of large models, exploiting multiple machines and multiple GPUs per machine. Currently, GPUs offer enough computational power to satisfy the needs of the state of the art models. However, there is a third point playing in the field: data privacy. In recent years, users and governments have started to be aware of this issue, becoming more important so that new and stricter regulations have been published. Even companies may want to shield themselves from any security leak that could happen in a centralised system. Then, the sight is moving to distributed systems where the data is not gathered into a central system, although it does not mean that a datacenter-scale approach could not also be distributed. The fast development of smart devices and their computational power and fast Internet connections like 4G, or even 5G in coming years, enables the approach to use them to train distributed models. The solution is not at the same scale of resources that a datacenter can offer yet, but the research and development of edge devices is making it feasible.

For these reasons, researchers are exploring the possibilities of different massively-distributed training designs. The new designs should offer scalability, ensure data privacy and reduce large traffic of data over the network. The main massively-distributed approach to large-scale training is Federated Learning [3].

1.1.2 Federated learning

Standard Machine Learning approaches require large amounts of data usually centralised in datacenters. In this approaches, there is only one device responsible for the training of the whole process. Instead, new collaborative approaches allow to train common models from different decentralised devices, each one holding local data samples. An example is Federated Learning[3]. It is a Machine Learning technique that provides a protocol to train a model in a massively-distributed way. It is not a fully decentralised technique as Federated Learning requires a central orchestrator. It is a central node that organise, distribute and control the training process and flow of data.

Federated Learning has been tested on very large network and with very complex models. It has been demonstrated flexibility to adapt to different types of models, along with a good scalability, achieving high quality results in a relatively low number of iterations. The nature of the data used can be of different types, from images to text. An example of the architectures tested are Convolutional Neural Networks to learn features from images. However, there are very few research about architectures used in Natural Language Processing tasks to test Federated Learning with text.

1.1.3 Natural Language Processing

Natural Language Processing (NLP) is an area of Artificial Intelligence that has become popular during the last decade. The growth in the number of posts, comments, reviews, tweets, etc., in social media makes available raw large texts with information about sentiments, ideas and desires of people. Collecting all that knowledge can be very useful in a wide range of applications. For example, companies could use the information to provide more useful product recommendation by targeting specific customers with good marketing in the commercial sector.

A central task in Natural Language Processing is the generation of word embeddings, i.e. representations of every word in a vector space, encoding the meaning of each word and the relationships between them. This task is usually performed by a Machine Learning model, such as Word2Vec[4], ELMo[5] and BERT[6] that provide high-quality vector representation of the words, capturing their meaning and context. These models are based in Deep Learning techniques like Artificial Neural Networks, requiring a large corpus of documents to use as input. These representations can be then be used to perform advanced analytics on textual data, such as reviews and social media posts. The larger and more complete the corpus is, the more accurate the representations and the analysis will be.

1.2 Problem statement

While the issue of privacy preservation affects lots of use-cases, the focus of this thesis is on one specific scenario: corpus sharing. It is represented by

a small number of organizations such as private companies or government agencies, each own a unique large corpora with specific information. These corpora might be skewed towards specific topics and might not be complete enough to individually train a high-quality language model. Thus, these organizations want to co-operate to build a unified model, but without sharing the contents of their corpora because it might be protected and regulated by privacy data collection agreements, or even containing sensitive user data from, for example, customers. Furthermore, if these organizations are private companies, they might not be willing to expose the full contents of their corpora to separate, independent companies, as these corpora might contain strategic information.

In particular, organizations and companies would benefit from training large, unified NLP models. The resulting models in this field are directly improved when they are training with a large rich corpus with multiple different topics. However, privacy regulations prevent traditional datacenter-level training on shared data.

1.3 Purpose

The purpose of this research is to implement and evaluate a distributed, efficient, data-private approach that allows a small number of organizations, each owning a large private text corpus, to train global word representations. Here it is the point where distributed decentralised training converges with NLP models. Having a model that can be trained without the need to centralised the data solves any data privacy issues that companies have, while they can share knowledge to collaborate between them without compromising data.

Thus, the main addressed research question in this study can be summarised as follows:

Is it viable to obtain a high-quality word vector representation under the perspective of a massively-distributed, efficient, data-private approach like Federated Learning where a certain number of organisations collaborate with their own private corpora?

It was not possible to find such previous work in the state of the art tackling

the problem stated when we started this project. However, a very recent paper[7] was published training BERT in a federated manner.

1.4 Approach

The most relevant state of the art techniques are reviewed, to identify the most suitable approaches to provide a proper answer to the previously stated research question. As starting point, Word2Vec is the baseline model in the scope. A reasonable continuation of the research would be to implement and test more advanced models to have a broader evaluation of the solution, and perhaps achieve better results. From the distributed perspective, Federated Learning is chosen as the scheme to follow along with Word2Vec. The resulting model is what we denominate Federated Word2Vec.

Several assumptions will be made to test the limit of the solution. By training an NLP model following the schema of a massively-distributed protocol, this study aims to provide an answer to the former research question. If the answer is negative and it is not viable to obtain the desired word representation, an analysis to discover the reasons why it does not work will be conducted.

The solution is tested and discussed from different points of view: a detailed analysis of the convergence of the training process; an assessment of the trade-offs between the number of organizations and size of the corpora; and semantic tasks to test the quality of the vector representations.

1.5 Goals

Following the approach explained in former Section, the research aims to fulfill a set of objectives. The objectives are divided, mainly, in three stages of the research. The first stage is focused on word embeddings models, gathering all data from state of the art. It allows to prepare a good implementation of the model that is faithful with what is described in the literature. Then, the second stage is a similar process is followed but in the area of Federated Learning. At the moment both implementations are finished, it is time to mixed them in the

final solution that is used in the experiments. The third stage is to experiment with the model, to obtain results, to analyse them and to revise the model to setup the baseline model and federated model metrics.

After the implementation and the experiments are finished, it is possible to provide a proper answer to the research question by looking at some key points as: the convergence of the models, the effects of different sizes of data and topic-specific data. Then, in order to achieve the main purpose of the research, the following goals should be completed:

1. Implement Word2Vec NLP model as a baseline score during the research.
2. Implement Federated Learning scheme to insert Word2Vec model in it.
3. Test the viability of Word2Vec plus Federated Learning working together.
4. Test Federated Word2Vec under different circumstances to test the limits of the solution. Provide the situations where the model converges and the reasons why it does not.
5. Collect truthful data in text format. The should have standard english level and different topics.
6. Find benchmark datasets to test the quality of the results.

1.6 Overall results

There are two main results in this research. The first result shows that it is viable to train an NLP model, in this study Word2Vec, with a massively-distributed technique like Federated Learning. The convergence results are similar to baseline Word2Vec and Federated Word2Vec when trained with the same data. In particular, Federated Word2Vec is benefited from training with larger data as it can process more data in less iterations.

The second result shows how the words, learnt from the model in the way of vectors, are placed within a distribution in the space. Words coming from topic-specific datasets find their own spot in the distribution, building clusters and keeping the spot independently of the size of the datasets or the content. Thus, it is stated the importance for organizations to cooperate, as cooperation

provides models that are not only globally good, but also locally better than locally-trained models.

1.7 Structure

By the end of this Chapter, the topic of this research is already introduced. It also summarised the question this research wants to answer, the goals, the research methodology used to achieve them and a brief overview of the most important results. Chapter 2 extends in deep the related work and history around the field in which the research is made. It also introduces the most relevant algorithms. Chapter 3 develops the research methodology followed in the research, detailing the steps, assumptions and simulated scenarios made. It prepares everything needed to understand before going into Chapter 4 where the experiments are described, graphs and metrics obtained. Finally, Chapter 5 indicates about the future work directions born from this work and deliberates about the discoveries of it.

Chapter 2

Related Work

2.1 The field of Machine Learning

Machine learning is the science and art of programming computers so they can learn from data [8]. It can be said that Machine Learning (ML) is a subset of the larger field of Artificial Intelligence (AI) and it also presents a close relation with statistics. While AI has the aim of demonstrating intelligence in a machine and statistics draws population inferences from a sample, Machine Learning is focused on the algorithmic perspective to find general predictive patterns [9]. Machine Learning focuses on teaching computers how to learn without the need to be programmed for specific tasks. In fact, the key idea behind Machine Learning is that it is possible to create algorithms that learn from and make predictions on data [10]. Examples of this discipline can be movie recommendation system [11] or mail spam filters.

As with all technologies, Machine Learning has some issues and obstacles which need to be addressed and no Machine Learning model could exist without them: **data** and **computation**. Both of them do appear in the definition, remarking the important role these key elements play in the discipline.

Machine Learning requires amounts of data carefully prepared to obtain a successful model. A model can not exist without data to predict. This was one of the main issues until, during the last decade, an explosion of data generation

occurred. It happened thanks to the daily access to browsers, social media, or applications that are used on many different devices from smartphones to smart televisions. This generates tons of data ready to be used by the Machine Learning algorithms created during the last century. For example, Random Forest [12] and Support Vector Machine [13] algorithms were discovered in 1995.

On the other hand, computers are the places where the algorithms are executed providing them with enough computational power to complete all their calculations. CPUs and GPUs performance has been increasing year after year, building computers powerful enough to run the vast majority of Machine Learning algorithms in a more than an acceptable amount of time.

Evolution in computational power and data available have been striking key points in the development of the field, making feasible any Machine Learning algorithm, although they become more complex every day. In exchange, their accuracy is highly improved, making worth the trade-off between complexity, resources required and accuracy.

Once the setup is prepared and the algorithms are ready to be trained, there are some previous task that can influence the performance of those models. High-quality data and preprocessing steps may have a bigger impact in the accuracy than the perfect tuning of the model parameters. For example, most Machine Learning systems works better dealing with tabular and structured data, where each row is one sample and each column one feature, with the number of columns being fixed.

Moreover, there are different algorithms with different characteristics. Depending on the data collected and the goal, the tasks can be to classify objects or recognised group patterns within the data. Different Machine Learning models can be categorised based on the feedback, type of information receive while learning, and the purpose, the desired end result.

Supervised learning consists of a training dataset organised in a set of input objects, usually vectors, and their corresponding desired output values, called labels. The main goal of supervised learning algorithms is to find a function that matches the input values with their labels. Then, this function can be used later to predict the output value from a given input. The most common use case for supervised learning in the classification problem.

On the other hand, unsupervised learning consists of a training dataset with only input objects, there are no output values available. Instead of finding a function to predict the output, the functions tries to describe the structure of the data and group this unsorted input objects. The most common use case for unsupervised learning is to cluster data.

2.1.1 Deep Learning

Deep Learning [2] is a specific subset of Machine Learning where the models, called Deep Neural Networks (DNN), are inspired by the human brain structure. It imitates the process of the human brain to define and recognise patterns while processing a huge amount of data for use in decision making. One of the main differences with Machine Learning algorithms is that traditional models analyse the data in a linear or non-linear unique kernel, while DNNs enable machines to process the data using multiple layers of non-linear transformations.

Larger data and more powerful computers made available the design of bigger and deeper DNNs. It seemed that the deeper the DNN, the better outcome. Convolutional Neural Networks (CNNs) [14] are a clear example. A CNN is a powerful DNN technique that is primarily used to solve difficult image-driven pattern recognition tasks and with their precise yet simple architecture. This trend is evidenced in the latest most popular CNNs architectures that achieved the best results in the DNNual software contest, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), where researchers compete to correctly classify and detect objects images present in the ImageNet [15] dataset. The architectures presented in the contest increased in size year by year: from AlexNet [16] in 2012 with 8 layers, passing through GoogleNet [17] in 2014 with 22 layers and right up to ResNet [18] in 2015 with 152 layers.

DNNs present a clearly differentiating characteristic that defines them: their scalability. Jeff Dean, Google Senior Fellow in the Systems and Infrastructure Group at Google, highlighted it in 2016 in a talk titled “Deep Learning for Building Intelligent Computer Systems” where he indicated that results of DNNs get better with more data and larger models, requiring longer times of execution.

In addition to scalability, another highlighted characteristic of DNNs is their ability to perform automatic feature extraction from raw data, also called feature learning. Yoshua Bengio, another leader in Deep Learning and professor at the Department of Computer Science and Operations Research at the Université de Montréal, cited this property in [19] where he commented that *"Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features"*

If both properties are mixed, the result is a model capable of excelling on problem domains where the inputs are analogues. In words of YDNN LeCun, director of Facebook Research and recognised as the father of the CNNs, *"Deep learning is a pipeline of modules all of which are trainable. Deep because it has multiple stages in the process of recognizing an object and all of those stages are part of the training"*. It describes the ability of DNNs of recognising features in images of pixels, documents of texts, video or audio. Although DNNs do not depend on tabular and structured data, they provide the best result when dealing with labelled data. In fact, most of the benefits of Deep Learning came from supervised learning.

2.2 Natural Language Processing

Each word and each sentence a human being generates during a conversation carries huge amounts of information. The topic of the speech or the complexity of the vocabulary chosen are two of the many variables that provide information and enrich the communication with natural language.

Trying to store and analyze all different structures, vocabulary, meaning and tone of every sentence is not a viable idea. Data generated from conversations, books or even tweets are examples of unstructured data in the form of text, this type of data does not fit neatly into the traditional tabular format where data is stored in rows and columns. In other words, it follows the design of relational databases. But most of the data, that is available nowadays, is not structured, hence it is difficult to manipulate. That is the reason why other research areas have evolved to mitigate these issues depending on the type of unstructured data, for example, graphs or text.

Natural Language Processing (NLP) is an area of AI that provides a machine the ability to read, understand, predict and derive meaning from human languages. This field is applied in different situations, the following applications are some among the big variety of them:

- **Sentiment analysis[20]:** it is the interpretation and classification of positive, negative or neutral emotions within a text. It provides information about customer's choices and their decision drivers. Since the surge of the Internet, sentiment analysis became more popular because people has the opportunity to find out about the experiences and opinions of others. Nowadays, more and more people are making their opinions available to strangers through social media, for example, sending tweets or uploading posts related to different topics, such as: politics, buying preferences, tourism; which makes this application very useful.

- **Machine translation[21, 22, 23]:** it is the process to translate information in one language into another by using a machine. The best-known application is Google Translate which is based on statistical machine translation (SMT) [24]. The idea behind it is to gather as much text as possible trying to find a parallel text in the other language based on the likelihood that the text appears in the other language.

- **Speech recognition[25]:** it is also known as Automatic Speech Recognition (ASR), or computer speech recognition. It is the process of converting a given speech signal to a sequence of words using an algorithm implemented in a computer program.

Some of these NLP applications can be possible after preprocessing the raw text so that any machine can understand it. It consists of representing the input word strings with numbers. The numerical representation should be semantically meaningful, capturing as much linguistic meaning as possible from each word. In the current paradigm of NLP, the dominant approach to achieve this is **word embeddings.**

2.2.1 Word Embeddings

One of the strongest trends in NLP at the moment is the use of word embeddings, which are vectors whose relative similarities correlate with

semantic similarity. Such vectors represent words as semantically-meaningful dense real-valued vectors, solving many of the problems presented in the one-hot encoding vector representation, for example, the sparsity of the vectors or the vocabulary size [26].

There are two different strategies to produce the same type of semantic model: count-based distributional semantics models and predictive neural network models. Both methods have the same goal and there is no qualitative difference between them, as several recent papers have demonstrated both theoretically and empirically the correspondence between these different types of models [27, 28]

A statistical perspective

At the beginning, NLP was focused on the mechanical way of extracting knowledge from a text or speech based on the keywords. The vectors resulting from this perspective are frequency-based embeddings. These techniques rely on a statistical approach and do not require training data to extract the most important keywords. However, this point produces a disadvantage in terms of selecting the important parts of a text because words or sentences that only appear once, which may be relevant in the text, maybe overlooked because of the fact of depending on a statistical perspective. Most of these techniques share the basic concept of word frequency. It consists of listing the words and phrases that most commonly appear within a text. It is a "*bag of words*" where aspects such as synonyms, grammar or structure are left aside.

- **Word Collocations and Co-occurrences [29]:** it is also known as N-gram statistics. It helps to understand the structure of a text thanks to the study of collocation, words that frequently appear together. On the other hand, co-occurrence is the number of times an N-gram appears in the text.

- **Frequency–Inverse Document Frequency (TF-IDF) [30]:** it is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. It is computed in two different scenarios: term frequency (TF) which measures how frequently a term occurs in a document; and inverse document frequency (IDF) which measures how important a term is.

$$\text{TF (i)} = \frac{log_2(Freq(i,j) + 1}{log_2(L)}$$

$$\text{IDF (t)} = log\left(1 + \frac{\text{number of documents}}{\text{frequency of t}}\right)$$

- **Rapid Automatic Keyword Extraction (RAKE) [31]:** the algorithm is based on finding and removing the stop words, provided previously in a list, in a text. The remaining words are called "*content words*". RAKE builds a matrix of the words counting the co-occurrences of each word to assign a score, the sum of the number of co-occurrences the word has with any other content word in the text. A keyword will be selected according to a threshold of T that defaults to one-third of the content words in the document.

Prediction based embeddings

A new approach has emerged in NLP during the last years. It is the cognitive way of processing text where the attention is put into the meaning behind the words and the context. It differs from the deterministic methods seen in the previous section because of the use of Neural Networks as their core technique. Statistics methods to build word vectors proved to be limited in their word representations until Mikolov et al. introduced Word2Vec [4, 32] architecture to the NLP community, meaning one of the first uses of a DNN in such a task. Word2Vec proved to be the state of the art method for word analogies and word similarities. It is well-known the example of words operation where the word *Queen* is obtained from the word *King*: $King - man + woman = Queen$. From a human perspective, it is a result quite logical, but from a machine perspective, it was considered a great advanced in the field.

However, Word2Vec is not the state of the art solution anymore. There are several new models that introduce novel techniques developed in Deep Learning, after the design of Word2Vec, to achieve better quality vector representation. Among these new models, two of the most popular are: ELMo[5] and BERT[6]. Apart from these two, there is another model, called GloVe[33], released months after Word2Vec. They address the same problem but from a different perspective, in other words, the objective of the DNN is the

same but the cost function and the weighting strategy are different[34]. Both achieve similar results and depending on the corpora one model can obtain better embeddings than the other.

After almost 5 years with no new deep contextualized word representation model presented, ELMo[5] appeared with the introduction of pre-trained bidirectional LSTM. The main difference is that ELMo word representations are functions of the entire input sentence based on three layers of representation, resulting in a word embedding that is a function of the entire input sentence.

Recently, after the release of ELMo, a new model has been released and it is the current state of the art approach: BERT[6]. This model introduces the concept of bidirectional training as the key aspect to achieve better results. It is possible by using masked language models. The architecture of BERT introduces the Deep Learning technique of encoders, resulting into a multi-layer bidirectional transformer encoder architecture.

2.2.2 Word2Vec

Word2Vec is based on the efficient Skip-gram model previously introduced by Mikolov et al. in [32]. It is an efficient method for learning high-quality vector representations of words from large amounts of unstructured text data. Although Skip-gram is used in Word2Vec, there is another model called Continous Bag-of-Words (CBOW) which follows a similar concept internally.

Continous Bag-of-Words Model

CBOW model is based on the architecture of the Feedforward Neural Net Language Model (NNLM) presented in [35] but removing the final non-linear layer and the projection layer is shared among all words in the vocabulary given [32]. The order of words does not affect the results of the model. The goal is to classify the middle, current trained label, word in a sequence of words.

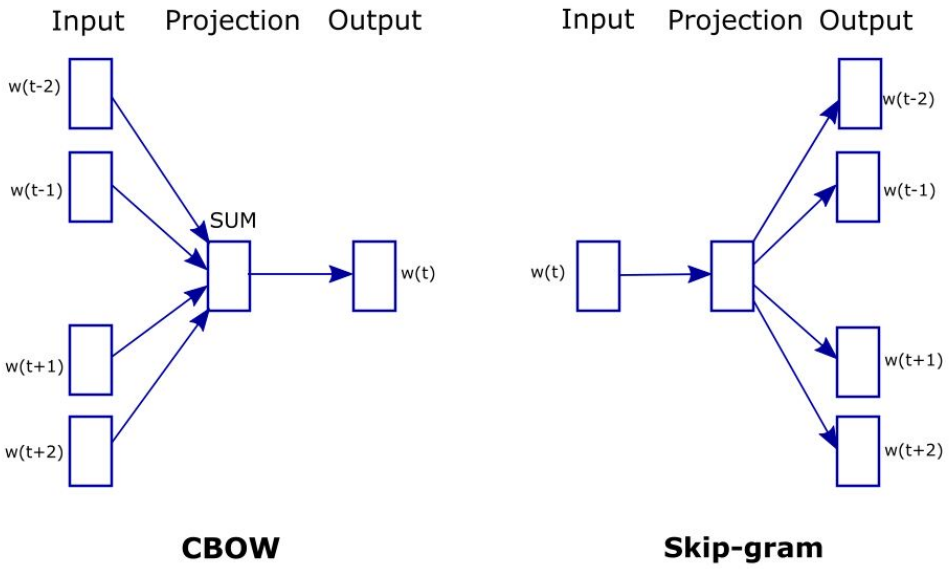


Figure 2.1 – The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. The image is based on the diagram that appears in [32].

Continuous Skip-gram Model

This architecture is similar in concept to CBOW, but it puts the focus on predicting the current word based on the context. In other words, it tries to find a word that is a candidate to appear in the context of the target word in the same sentence. It requires to prepare the data in *(target_word, context_word)* pairs where the current word is the *target_word* playing as input data and *context_word* as label data.

As mentioned in [32], the number of context words to be labelled depends on the window size width to define the context to be analysed. If the range in increased, the quality of the vector is better as the context is more specifically defined, but the computational load also increases. Since the more distant words are usually less related to the current word than those close to it, it is often given less weight to the distant words by sampling less from those words. The complexity of the architecture is proportional to:

$$Q = C \times (D + D \times log_2 V)$$

where C is the maximum distance of the words, D the embedding size, which is corresponded with the number of columns of the weight matrix, and V is the output layer dimensionality, which is equal to the lenght of the vocabulary. In the experiments run by [32] the final parameters selected were 10.

Furthermore, it was presented in [4] two additional improvements to the Skip-gram model. Both are based on strategies to speed up the process to reduce the convergence time and computational load.

Hierarchical softmax

Hierarchical softmax is an approximation technique to the full soft-max probability distribution function, it was published by Morin and Bengio [36]. It is allowed by the approximation to compute only $log_2(n\_nodes)$, compare to the computation of the total number of nodes of a usual softmax. It structures the output layer in a tree diagram where the leaves are the words representing the relative probability of its child node.

The binary Huffman tree [37] was used by Mikolov et al. in their experiments. The main outcome was a considerable effect on the performance, resulting in faster training.

Negative sampling

Negative sampling[4] uses P_n as the noisy distribution to select the negative samples. The next formula is the loss function whose output is provided to an optimization algorithm. It is defined as follows:

$$\log \sigma({v'_{wo}}^{\top} v_{wI}) + \sum_{i=1}^{k}{}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{wo}}^{\top} v_{wI})\right]$$

The task is to differentiate target words from noise using a simple logistic regression, where the noise is represented by k negative samples for each data sample. It means that there are true *(target_word, context_word)* pairs mixed with false *(target_word, false_context_word)* pairs. The regression

distinguishes true pairs, building quality vector embeddings, from false pairs, speeding up the process by reducing computation time. The experiments in [4] indicated that a good parameter number of k negative samples per batch is 2 to 5 for large datasets.

Furthermore, Negative Sampling is an idea that comes from the statistical model Noise-contrastive estimation (NCE), first introduced by [38] and then reformulated by [39]. The basic idea is to train a logistic regression to discriminate between the observed data and some artificially generated noise (same process followed in negative sampling), using the model log-density function in the regression nonlinearity. Then, NCE is based on the reduction of density estimation to probabilistic binary classification. It was used in [40], allowing them to fit models that are not explicitly normalized making the training time effectively independent of the vocabulary size.

2.3 Distributed Machine Learning

The demand for artificial intelligence has grown significantly over the last decade and this growth has been fueled by advances in Machine Learning techniques and the ability to leverage hardware acceleration. However, in order to increase the quality of predictions and render Machine Learning solutions feasible for more complex applications, a substantial amount of training data is required. It is specially important when training large Neural Networks because DNNs grows exponentially with the number of parameters.

The only way to solve the problem is to scale the computational resources. Scalability is the ability to handle an increased workload by repeatedly applying a cost-effective strategy for extending a system's capacity [41]. This definition derives into two alternatives: scale up, it consists of replacing your resources with something more powerful. It is possible until you reach the technology power limitations of an individual component; scale out, it takes the resources available and replicates it to work in parallel. This has the effect of increasing infrastructure capacity roughly linearly.

Nowadays, parallelization of the Machine Learning workload has become paramount to achieving acceptable performance at large scale, GPUs and accelerators are now more common in major cloud datacenters [42]. It happened because of different reasons around scalling out approach [43]: the

first is the generally lower equipment cost; the second is the resilience against failures because the distributed architecture includes back-up nodes, or even without them it is possible to keep the execution; the third reason is the increase in aggregate I/O bandwidth compared to a single machine. Machine Learning tasks are highly data-intensive and the ingestion of data can become a serious performance bottleneck. Augmenting the number of nodes also augments the individual I/O systems, hence more data fed in the same amount of time.

There is a common point of view around the definition of the architecture of a Machine Learning system. It consists of two main phases: training phase, it involves training an Machine Learning model by feeding it with a large amount of data in order to update properly the model based on the data it has seen; prediction phase, it is the time when the model is deployed and put in production. The first phase is typically more computationally intensive and requires the availability of a large dateset, while the second phase can be performed with fewer resources.

It is possible to extrapolate this partition to the distributed architecture and distributed training of an Machine Learning algorithm. There are also two fundamentally different strategies to address the problem of splitting the task across multiple machines, parallelizing the data or the model:

- **Data-parallel** approach consists of creating partitions, as many as number of machines in the system, of the whole dataset and distribute them across all nodes. Then, the same Machine Learning algorithm, independently of the Machine Learning technique selected, if all share an independent and identically distribution assumption over the data samples, is applied to the different partitions in each node. The model is available to all nodes either through centralization or replication.

- **Model-parallel** approach consists of processing datasets in each node which works on different subparts of the model. At the end, the resulting model is the aggregation of all model parts coming from the nodes. This approach is not available for all Machine Learning algorithms because their parameters may not be split up. In this case, the strategy applied consists of training different instances of the same model and ensemble all the models at the end.

From these approaches, another element can be distinguished through the process which is the existence of a central organizer or main controller node

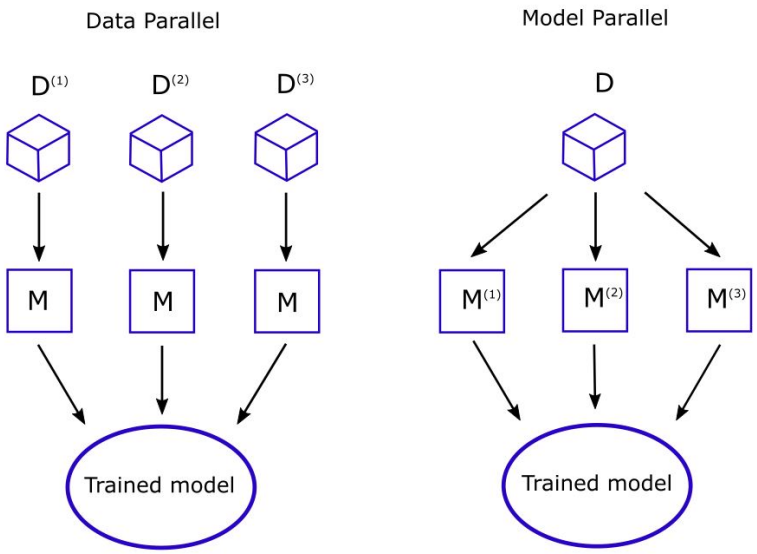


Figure 2.2 – Parallelism in distributed Machine Learning: Data-parallel and Model-parallel. The figure is based on the image in [44].

or the totally decentralized version.

2.3.1 Decentralised Machine Learning

There are extreme scenarios where a distributed Machine Learning approach does not count on an orchestrator to supervise all the process and to collect all the data shared by the nodes after each training iteration. In those scenarios, the system does not have a central gate-keeper, so no party can have full control of the process over the protocol. Additionally, all members of the system present the same running model, without the need to trust each node independently. Each node has extra powers and privileges.

Therefore, a fully decentralised Machine Learning approach provides transparency and independence. However, the security breach is not identified in just one node, but spread in all the nodes, which might be exploited by malicious activities to leak the information.

These techniques also present useful properties such as scalability and flexibility, without the need of central control. A peer-to-peer network protocol

can scale up under any circumstances until an unlimited number of devices, while a centralised version needs the agreement of the central node to allow a new node to belong to the network. Then, decentralised techniques are rapidly adapt to changing users, and also being prepared for extreme failure scenarios.

There are different decentralised approaches, but the main results in the field of decentralised machine learning is the gossip learning protocol[45]. There are also more examples like decentralised clustering approaches [46]

2.3.2 Gossip Learning

Gossip Learning introduced in[45], is a simple and effective asynchronous protocol based on gossip communication approach, which relies in a total decentralised communication without the need of an orchestrator. Gossip Learning is still a field to be fully discovered by the community. Multiple published researches have shown that gossip learning is very flexible and it has been successfully applied to different kinds of machine learning problems and algorithms, for example, binary classification with support vector machines [45]. Recently, extended work was done about this specific classification problem and gossip learning[47], clarifying the use of these techniques with Machine Learning algorithms.

The core concept of gossip learning is to have a set of models, which move through the network, being shared by the nodes. When a model visits a node, it is trained on its local data and merged with the last model that visited the same node. In this way, the models can quickly acquire knowledge from a large number of nodes, until they all converge to a fully-trained global model. The scheme of the protocol is shown in Algorithm 1.

2.3.3 Federated Learning

Federated Learning can not be considered a decentralised protocol as it is Gossip Learning protocol. Although both protocols try to solve the same issues, they do it from a different perspective. Federated Learning requires a central orchestrator. It is a central node that organise, distribute and control the training process and flow of data. Thus, it can be called a massively-distributed protocol.

Algorithm 1 Gossip Learning scheme

$currentModel \leftarrow InitModel()$
$lastModel \leftarrow currentModel$
loop
  Wait(Δ)
  $p \leftarrow RandomPeer()$
  $Send(p, currentModel)$
end loop
procedure ONMODELRECEIVED(m)
  $currentModel \leftarrow CreateModel(m, lastModel)$
  $lastModel \leftarrow m$
end procedure

Federated Learning was first introduced by McMahan et al. [3]. They carried out the training of deep Neural Network fed with data from a large network of edge devices, without having any of the data leak out from its owning device to the central node, and exploiting the spare computational power of these machines. This approach has been tested on very large network, and with very complex models. It has been demonstrated flexibility to adapt to different types of models, along with a good scalability, achieving high quality results in a relatively low number of iterations.

In Algorithm 2, it is explained the two main procedures that take place in parallel. The protocol is divided into two groups, the orchestrator or central node which controls the execution receiving the gradients and returning the updated model back to the second group, the external nodes. Those are the devices where the data is stored. They also have to calculate the gradient, moving most of the computational load out from the central node.

The protocol can be performed in two different ways: the first one is based on sharing the gradient in each iteration; while the second one allows the external nodes to update the weight matrices for several iterations until the learning gather by the nodes is shared to the central node. In this second approach, instead of sharing the gradient, the weight matrices are shared to the central node. It means that there is a trade-off between the number of updates to the central node in exchange of reducing the traffic network.

Algorithm 2 Federated Learning Protocol scheme

procedure SERVERLOOP(())
  loop
    $S \leftarrow RandomSubSet(devices, K)$
    for all $k \in S$ **do**
      $SendToDevice(k, currentModel)$
    end for
    for all $k \in S$ **do**
      $w_k, n_k \leftarrow$ ReceivedFromDevice(k)
    end for
    $currentModel \leftarrow$ WeightedAverage(w_k, n_k)
  end loop
end procedure
procedure ONMODELRECEIVEDBYDEVICE(model)
  $model \leftarrow model+$ Update($model, localData$)
  SendToServer($model$, SizeOf($localData$))
end procedure

Chapter 3

Research Methodology

3.1 Word Embeddings and Federated Learning

The previous chapter discussed two different approaches to train a distributed machine learning model: Gossip Learning and Federated Learning; along with the different techniques developed in the recent year in the Natural Language Processing field.

In this research, it was decided to use a distributed approach to train a classic NLP model to provide an optimal word embedding representation of a text. After deep research through the literature, it was not possible to find any previous research about an NLP model trained in a distributed environment. First of all, it is needed to engineer a solution that merges both concepts. To achieve the goals explained above, given the lack of reference architectures for this kind of projects in the literature reviewed, a solution was designed from the ground up. The main components of this architecture are **Federated Learning**, a well-known distributed machine learning technique, and **Word2Vec**, a popular word embedding model.

By training a Word2Vec model following the Federated Learning protocol, this study aims to provide an answer to whether it is possible to achieve a high-quality vector representation of text data without centralising the dataset, or if it is not viable, analyse the reasons why it does not work.

3.1.1 Federated Learning choice

As was mentioned in Chapter 2, the interest in distributed machine learning models is currently increasing. The trend in popularity and utility of distributed protocols is the result of a combination of issues around the society digital transformation.

Among the different distributed techniques mentioned in Section 2.3, Federated Learning is the approach selected to carry out this research. The choice is based on the continuous improvements made around it and the future scalability and impact in the training strategies. Since the introduction of Federated Learning in [3], many works have relied on this approach. Not only theoretical research, but also frameworks and software libraries dedicated to facilitating the use of Federated Learning. An example is TensorFlow Federated * which adds new functionalities to the core of TensorFlow [48] to ease the process of adopting or integrating Federated Learning in machine learning projects.

Moreover, Federated Learning addresses the privacy concerns as the data is not shared, it stays in each node and the information transferred through the network is purely the gradient of the Neural Network. It avoids sharing raw information from the training data, in addition, the total size of the data transferred is, generally, less than the total dataset size. This makes Federated Learning a suitable protocol to fulfill the goals of the project.

The last point is the existence of a central node which orchestrates all the traffic and makes all the decisions, it is the only common point that all the nodes share. Having the central node can facilitate the inclusion of extra safety measures to bulletproof the connections against possible cyberattacks.

Most of the experiments are run using **FederatedSGD** algorithm [3] which transfer the gradient from all the external nodes to the main node in each iteration, as shown in Alg. 3

* https://www.tensorflow.org/federated

Algorithm 3 Baseline Federated SGD

In **Central Node:** $main\_Model \leftarrow Init\_Model()$
In **External Node:** $datasets_n \leftarrow Load\_Dataset(num\_nodes)$

procedure CENTRAL_NODE($nodes$)
  loop
    for all $k \in nodes$ **do**
      $Send\_Model\_To\_Device(k, main\_Model)$
    end for

    Wait until $External\_Node$ ends iteration

    for all $k \in nodes$ **do**
      $g_k, n_k \leftarrow Received\_Gradient\_From\_Device(k)$
    end for
    $\nabla f(w_{main\_Model}) \leftarrow Agreggate\_Gradient(g_k, n_k)$
    $w_{main\_Model+1} \leftarrow w_{main\_Model} - \eta \nabla f(w_{main\_Model})$
  end loop
end procedure
procedure EXTERNAL_NODE($model$)
  $w_k, updated\_model \leftarrow Update(model)$
  $g_k \leftarrow \nabla f(w_k)$
  $Send\_To\_Central\_Node(g_k)$
  Wait until $Central\_Node$ sends model
end procedure

3.1.2 Word2Vec choice

Word2Vec was the first Neural Networks model trained to generate a high-quality word embedding representation. It started the road of using Machine Learning models in Natural Language Processing, then it was followed by other new models such as: GloVe[33], comparatively, there is no better model between both, only the perspective to generate the vectors is different; ELMo [5], it provides better results thanks to the use of bi-directional Recurrent Neural Nets, resulting in a more complex model; and finally, BERT [6], which is the state of the art solution to extract word embeddings from a text. It uses autoencoders instead of RNNs and keeps the complexity of the model in comparison to Word2Vec.

Since the introduction of Word2Vec, other models have been developed which outperform it. However, it provides high quality word representations yet. Furthermore, it presents a number of advantages that make it preferable for this project, being still an acceptable model with low complexity compared to its succesors. For example, training BERT takes much more time to be trained, compared to the size of Word2Vec. If a more complex model was selected, it could lead to finding some issues derived from the baseline model and not inherent to the research, interfering with it. It could also limit the size of the simulation, reducing the number of external nodes to fit in a single GPU. These reasons make Word2Vec the ideal baseline model to first create an empirical research combining NLP models with Federated Learning.

Word2Vec uses Noise-Contrastive Estimation to speed up the process of calculating the loss, providing a computationally efficient approximation of softmax function. Instead of using a softmax function in the output layer to predict the output word, a binary classification is used. In other words, it converts a multinomial classification problem into a binary classification problem. NCE is slightly customized in [4]. The flow diagram in Figure 3.1 represents the architecture of the solution that has to be replicated.

However, it requires more operations between matrices before the computation of the loss. So, a cleaner and faster implementation of the architecture can be elaborated thanks to the TensorFlow API * where there is a predefined function to compute *NCE loss*. It results in a more legible and elegant solution as it can be seen in Figure 3.2, where it is represented the architecture of the

* https://www.tensorflow.org/api_docs

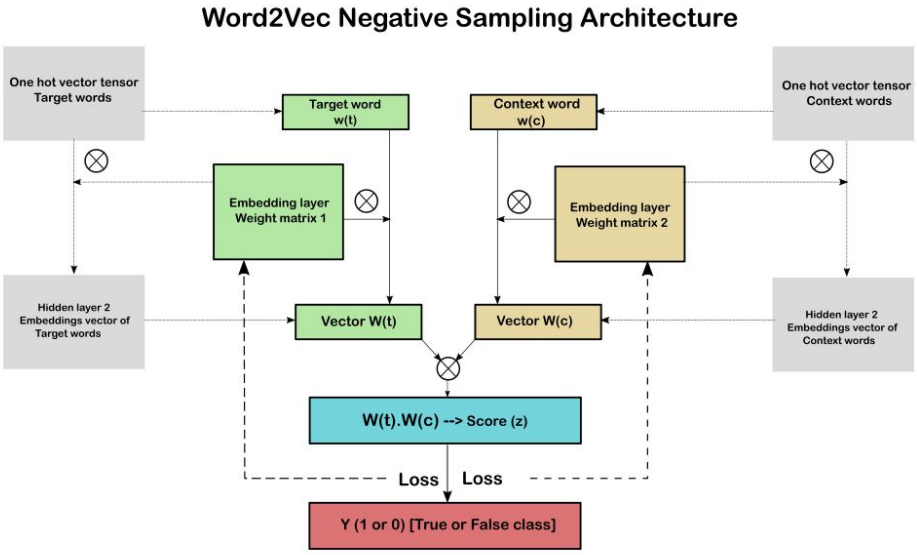


Figure 3.1 – Word2Vec Negative Sampling Architecture in a flow diagram in TensorFlow. Diagram based on [49]

Neural Network with fewer operations.

It is important to notice that, after training, the model itself is not directly used for NLP tasks. Rather, matrix U is extracted and used in isolation. This matrix is interpreted as the embeddings of the vocabulary given, where each row represents a word from it. For example, computing the cosine similarity between rows provides information about which words appear in the same place and in similar contexts. On the other hand, matrix V represents the context words, one word in the vocabulary per column. It is, to all effects, another embedding matrix. So, matrix V could also be used as embedding matrix. There are some techniques that either concatenate each row of U with the corresponding column of V, or join them in order to get even better embeddings. However, matrix U alone already provides decent embeddings and is the easiest way to use the output of Word2Vec.

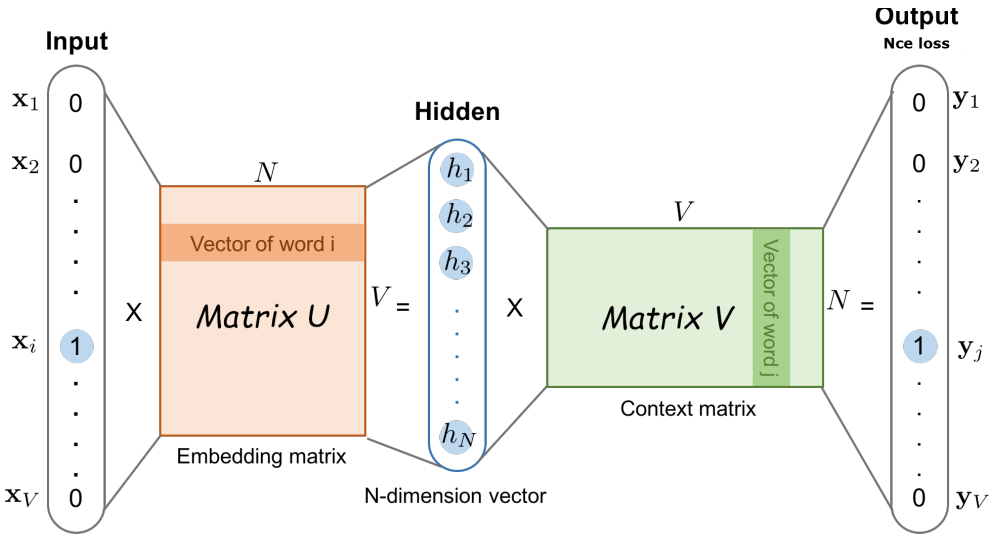


Figure 3.2 – The skip-gram model architecture with only two matrices: U matrix gathers the meaning of target words; and V matrix gathers the meaning of context words. The output layer uses nce_loss function instead of soft-max. Image based on [50].

3.1.3 Distributed Word2Vec

The final aim of this research is to test the viability of Federated Learning with NLP models, starting to experiment with Word2Vec. While Federated Word2Vec could be applied in many different circumstances, this project mainly focuses on one particular scenario, as introduced earlier. In this scenario a small number of organizations contribute with their data, each owning a large private text corpus, to train a global word representation. The architecture of the model follows a distributed, efficient, data-preserving approach to meet all the requirements. Figure 3.3 illustrates a small concept with 3 organizations.

Each organization owns a private dataset, which means that words that appear in the corpus of one organization may not be present in the one of another. This represents a preprocessing problem as the size of the matrices of the model depends on the vocabulary size, and so it must be common for all the organizations so that the gradients can be aggregated. Moreover, each word is represented by a row, so the order of the vocabulary, which is the same as the order of the rows, must also be common so that the updates of the gradients are made on the same words.

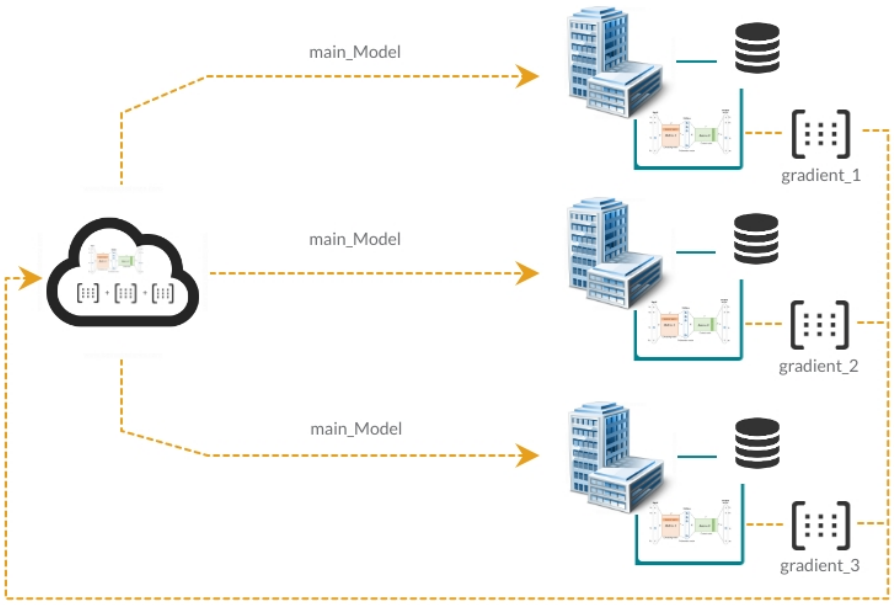


Figure 3.3 – Schema of Federated Word2Vec with 3 participating organizations. Notice that only the gradients are exchanged by the organizations and that each one has an independent private dataset.

3.1.4 Building a common vocabulary

Building a common vocabulary is an issue that the organizations should resolve in the first stage of the process. It is important to preserve the privacy of the content of the text, so it is not possible to centralise the text, preprocess it and return the processed text back to the organizations. Therefore, organizations must agree on a fixed vocabulary size N, a minimum threshold of occurrences T. Each organization must provide a list of their top N words that appear in their respective texts surpassing T occurrences. The privacy is preserved because the organizations only share a list of isolated unordered words with the orchestrator.

However, the most probable scenario is that all lists contain different words in different positions, hence appending the lists does not provide the final vocabulary. There are two operations within sets that can be applied to get the final vocabulary:

- **Intersection:** the result is the elements that belong to all vocabulary lists. The final vocabulary is shorter because words that belong to the specific vocabulary of a organization are excluded. The vocabulary is not rich in diversity and only common words are trained. On the other hand, the convergence of the model is faster because there are fewer words and they appear more times.

- **Union:** the result is all elements of all lists. The final vocabulary is larger than the fixed size vocabulary, but all organizations keep their words trained. This approach requires more time to converge because many words appear only in certain datasets. However, the meaning of the words and knowledge return to the organizations is enriched.

In the research, this process is simulated during a preprocessing phase, using the union approach to analyse the behaviour of words that only appear in categorised datasets.

3.2 Data Collection

Training any machine learning algorithm requires data. This should be structured, high-quality, truthful and available in sufficiently large amounts. This research is focused on NLP techniques so the data used comes from raw text. Text is usually found in an unstructured shape. It must be preprocessed and organized, for example, in $features-label$ vectors in supervised learning, or just in feature vectors in unsupervised learning, which is the type of learning used in Word2Vec, and therefore in this research.

There are no benchmark datasets in the state of the art that most of the community uses, as it happens, for example, in image processing, where ImageNet or MNIST are regarded as a standard. The most frequent approach is to gather data from social interactions such as tweets and Amazon reviews*; or collect articles from Wikipedia and newspapers. While the first group is oriented to sentiment analysis tasks, Wikipedia and newspaper articles work better to extract meaning and grammatical structures from a language, as written, semi-formal text is better suited to this task.

* http://jmcauley.ucsd.edu/data/amazon/

Thus, a Wikipedia compilation of articles from different categories was collected because this type of data meets, in general, all the conditions to fulfill the purpose of this research.

3.2.1 Wikipedia dumps

Wikipedia offers free downloads of all the content published on their website. The Wikipedia database can be used for all purposes, such as queries, offline use, personal use or informal backups. All text is protected* under the multi-license of CC-BY-SA and GFDL.

Wikipedia provides daily dumps from their database in different data formatssuch as XML, HTML, JSON and raw text. The compression format is bz2, resulting in a 16 GB file with all the text content published in Wikipedia, other types of data such as images, citations and references are excluded in these dumps. There are different languages offered, but most of the articles are in English and the research is conducted in this language, so only the English dump version was downloaded.

Wikipedia also makes available partial dumps of the database. The file downloaded is smaller but less diverse. For the initial low-level experiments and preprocessing tests, a partial dump was downloaded with a compressed size of 180 MB and real size of 256 MB. The goal was to develop and tune the model in a controlled environment with a short amount of data. For the final experiments, the whole dump was download with the status of the database as of the 1st of April.

Extract raw text

Bz2 is a compressed file format that in order to facilitate the extraction and organization of the data, there is available a script in Github† which extracts and perform a basic clean of the data, organising it in different formats such as html, json or raw text. It also adds the option to apply personal templates to extract specific information. The extraction is made in different files of the

* For more information about licensing and public use, refers to Wikipedia copyright

† The source files are available in `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor` website.

desired size where all the Wikipedia articles are concatenated until the size is reached and a new file is created.

The information stored about the articles is the title, the content, url, id and revised id. To speed up the extraction and avoid extra steps, the scripts were modified to store only the text of the articles.

Moreover, the script also allows to include a filter to extract exclusively the pages under the categories indicated in the arguments. This makes it easier to create specific datasets to simulate the diversity of topics that appear in different organization sectors.

Cleaning raw text

Once the datasets are collected, before feeding the data to the model, a preprocessing step is needed to prepare the data. This procedure is divided into several phases to avoid preparing the data in every training.

- The first phase is to clean the text to remove special characters such as punctuation marks, brackets and slashes among others. Most frequent english contractions were expanded to gather similar meaning words in the same expression. The option to remove stop words, which occur millions of times in very large corpora, was considered but eventually not implemented because based on existing literature and early experiments, was not expected to provide any benefits. On ther other hand, words that appear less than 10 times are removed because it will not be possible to obtain a good vector representation for them.

- The second phase is tokenization. It is done by using the pre-trained Punkt tokenizer in the well-known NLTK library*. After applying the tokenizer, the result is a list of all separated words that appear in the text.

- The third phase is to assign numerical ids to the words, given a predefined vocabulary and transform the words into their numerical representations. The ids are assigned and sorted by the number of occurrences. In this step, the text is transformed into numeric values that can be fed into the Neural Network.

* API information in https://www.nltk.org/

3.3 Experimental Setup

The experiments consist of a simulation of a real scenario where a group of 10 organizations want to collaborate to generate a word embedding representation of their corpora. The simulation is performed sequentially on a single machine and on a single GPU. It is coded in Python 3* language and using the very well-known TensorFlow† library as the end-to-end open source platform for machine learning in the research. In particular, it was used as the second major version of TensorFlow, released during 2019 because of some constraints simulating more than one model training in the same machine using the previous release.

The network traffic is not simulated to simplify the process, so all nodes that are trained in the same iteration are updated sequentially. However, the size of the information that would be sent through the network is calculated and analyse.

All the experiments are made with the same hyperparameters, unless otherwise stated in the detailed description of the results in Chapter 4. The main hyperparameters to take into account during the process are specified in 4.1.

These simulations required a huge amount of computational resources as 10 models, one per organization, need to be trained at the same time in the same machine plus the central node, while in a real scenario the workload would be spread among 11 machines. In order to complete the experiments, the machine used has the following specifications:

- **CPU:** 2x Intel Xeon 4214, each: 12 cores (24 threads), 2.2 GHz base clock, 3.2 GHz max turbo boost, 16.5 MB cache
- **RAM:** 192 GB (12x 16 GB DDR4 ECC registered, running at 2400 MT/s)
- **GPU:** 2x NVidia Quadro RTX 5000 with NVLink, each: 3072 CUDA cores, 16 GB GDDR6 memory

* https://www.python.org/ † https://www.tensorflow.org/

- **Storage:** 2x 1TB M.2 SSD

Although there are two GPUs available, only one GPU in a single thread was used during the experiments.

3.3.1 Datasets by categories

organizations and users expect a model that recognises their particular terminology and style. Having many nodes means more diverse topics and a wide range of vocabulary, making more complex the fast convergence of the model.

One experiment focuses on the transfer of knowledge between nodes to the central server in a scenario in which each node has specific words from a topic. If common words are trained jointly while topic-specific words are not negatively influenced by other nodes, it can be concluded that the topic-specific knowledge is kept in the model.

The Wikipedia Extractor script is used with the component to filter categories to prepare 5 different datasets divided by topic, those are: biology, history, finance, geography, and sports. Although the themes are quite specific, some articles can appear in more than one dataset because of the distribution of the Wikipedia tree of categories. So, if an article is tagged with the biology category, it is included in the dataset of biological content.

3.3.2 Datasets by size

In a real scenario not every organization owns the same amount of text. If a corpus is larger, it will have more words and more extensive vocabulary, and the number of iterations needed to analyse the whole dataset will be greater. Given this, it is interesting to analyse the behaviour of the model when some nodes have shorter corpora, as the model could tend to specialise in learning only the words of those nodes. The reason is that the shorter the corpora, the lesser number of iterations are required to complete one total epoch. Hence, the specific words from that node are updated more times than other nodes. One possible solution is to weight the gradients, at the moment of collecting them in the central node, to make a shorter impact in the learning of the main model.

Reusing the datasets divided by categories, the dataset of finance was reduced to half of its previous size while the rest kept their size. Finance was selected because it is the category with the most specific vocabulary.

3.4 Evaluation metrics

Word embeddings are a real-value vector representation of words. Learning a high-quality representation is extremely important for a wide range of applications as it was discussed in Section 2.2. Although there is no standard metric to evaluate the quality of the model as there is for a classification (accuracy) or regression (root mean square error) problems, there are various evaluation metrics that help to test word embedding models.

A high-quality word embedding is represented by the following properties:

- *Non-conflation* [51] is the ability to provide specific properties of a word given their different local contexts, e.g, verbal tenses.
- *Robustness against lexical ambiguity* [51], all senses of a word should be represented.
- *Demonstration of Multifacetedness* [51], the linguistical properties of a word should contribute to its final representation.
- *reliability*[52] and a good and spread *geometry*[52].

In order to check if the word embeddings from a model fulfill all of the previous parameters, a good evaluator needs to be defined. There are two types of evaluators[53]: intrinsic and extrinsic. However, both of them should gather all of the following characteristics[53]: good testing data, comprehensiveness, high correlation, efficiency and statistical significance.

In this research, only intrinsic evaluators were selected, namely word similarity, word analogy and concept categorization, because extrinsic evaluators[53] require more expensive resources and time, for example, perform Part-Of-Speech-Tagging, named-entity recognition or sentiment analysis. The three chosen intrinsic metrics were implemented based on freely-available

sources [*]. It performs severtal tests in different well-known datasets that are explained in more detail in Sections 3.4.2, 3.4.3 and 3.4.4.

3.4.1 Validation Loss

Loss is computed during the training phase to calculate the model error and provide it to the optimization algorithm chosen, which in this research it is Stochastic Gradient Descent (SGD) optimizer. It is known that loss is not a good indicator to discern between one model or another, because the model with the lower loss may not be the one with best evaluation metrics. Although it is not desired to choose a model based on this metric, loss is still useful to analyse in order to evaluate the convergence of a model, therefore how good the ANN is capturing the knowledge of the words.

In Federated Learning, the central node is the owner of the model but it has no data available. Therefore, it is not possible to calculate a centralized training loss. Hence, the training loss is partially calculated node by node, providing information about the convergence of each node. It may be used to detect different speeds of convergence or some stuck nodes.

However, there is still no information about the aggregated model itself. So, just for the purpose of this research during the simulation of the real case scenario, every organization prepared a small validation dataset with a size of 10% of the training dataset. All portions are collected and concatenated in the central node at the beginning of the training. In this way, it is possible to compute the validation loss of the model at the central node, to have a valid metric to study the convergence of the main model and to compare it with the validation loss of the baseline Word2Vec model.

Finally, it is important to notice that a common use case is to use $NCELoss$ during training, and calculate the full sigmoid loss for evaluation or inference. This metric is not calculated every iteration so the impact on resources and execution time is minute.

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

3.4.2 Similarity

The batch of standard similarity evaluation metrics is compound by a total of 8 different datasets of words pair taken from state-of-the-art literature. Each dataset provides pairs of words, with each pair having a similarity score. These scores are compared to the ones obtained using the model embeddings. There are some datasets that provide exclusively similarity pairs, for example, *cat-kitten*. While others have a more specific rating, where there are pairs of entities that are associated but not actually similar *Freud-psychology* have a low rating. The output metric is the Spearman correlation between the similarity score of the model and the human rated similarity of word pairs. The higher the correlation between ground-truth datasets and trained model, the better the final evaluation score of the model.

The list of datasets are the following: MEN[54], SimLex999[55], WS353[56] by default, WS353 with relatedness, WS353 with similarity parameters, RW[57], RG65[58] and MTurk[59].

3.4.3 Analogy

The batch of standard analogy evaluation metrics is compound by a total of 2 different datasets of words groups. Each dataset provides pair of words that are connected by some an analogy, for example, in Google's dataset the groups are formed by *capital-country* $\rightarrow$ *Athens-Greece* to *Baghdad-Iraq* or *adjective-adverb* $\rightarrow$ *amazing amazingly* to *calm-calmly*; and the trained model should be able to capture such analogy. The output metric is the summarized accuracy across categories.

The list of datasets are the following: Microsoft Research MSR[60], Google Analogies * by Mikolov et al. and SemEval_2012_2 †

* http://download.tensorflow.org/data/questions-words.txt † https://sites.google.com/site/semeval2012task2/

3.4.4 Categorization

The batch of standard evaluation metrics is compound by a total of 6 different categorised datasets of words groups. Each dataset has its own categories from vegetables to means of transport. If the low-dimensional embeddings generated by the trained model exhibit word clusters that closely match these ground-truth categories, then the model gets a higher score. The purity is the output metric to be compared within cluster in the dataset and cluster obtained from the model.

The list of datasets are the following: AP[61], BLESS[62], Battig[63] and ESSLI[64].

3.4.5 Visualisation with PCA

A word embedding is an n-dimensional vector and it is hard to obtain a good visualisation. In the case of Word2Vec, it is not only a vector, but as many vectors as the size of the vocabulary. It is interesting to display the embeddings to analyse different properties, for example, their neighbourhoods. They are detected base in different available distances such as cosine and euclidean.

TensorFlow released a visualisation tool, called Embedding Projector [*], to display the embedding matrix in a 3D space. To achieve it, there are 3 different techniques available to reduce the dimensionality of the vectors: Principal Component Analysis, t-SNE and UMAP. PCA was used throughout the project to perform a qualitative exploration of the embeddings based on some example plots provided in the results.

[*] https://projector.tensorflow.org/

Chapter 4

Results and Analysis

The experiments presented in this section do not have the goal to obtain the best possible results. Instead, the goal is to provide fair comparisons between baseline Word2Vec and Federated Learning Word2Vec, and variations of it, under the same circumstances and parameters. Thus, the execution times are limited at a certain number of total iterations.

It is important to notice that execution times can not be directly compared because the training of the external models is made in sequence in this simulation, while in a real scenario the training is in parallel, spending, in our setup, different amount of time in training one, two or n same models.

Moreover, when a comparison is made, it is always between models trained under the same setup and resources for this research. It would be unfair to compare the models trained in this research with results obtained in other studies due to limited computational resources.There is no hyperparameter tuning to find the best configuration and the training is performed on a smaller scale compared to modern state-of-the-art results, in terms of both training time and dataset size.

4.1 Common parameter setup

Following the reasoning from the previous paragraph, a hyperparameter search was not performed in depth because the aim of the study is not to compete against state of the art results, but to understand the effect of Federated Learning on Word2Vec. The hyperparameters were set to sensible values and kept constant throughout this study. All the experiment were run under the following conditions:

- Every node and the baseline model receive the same amount of samples per iteration, **fixed batch size** of 2048 samples. It is important to notice that one iteration of Federated Word2Vec processes as much data as N iterations of centralized Word2Vec, because each of the N nodes processes one batch in parallel during each iteration. In this research, it is expected to use a large batch size in order to better exploit the hardware resources and be able to fit more epochs within a limited amount of wall-clock time.

- **Window size** defines the number of context words that can be selected from the target word. For example, if the window size is 2, the context words would be the two closest words on the left and on the right, a total of 4 context words. The window size should not be too small in order to capture the context of the word and differentiate it from other words. The window size is fixed at 8 so that most of the relevant context of a word is captured.

- Sizes of around 200 and 300 dimensions for the **embeddings size** are common in the literature, and in this study we fixed it to 200, as this would reduce the memory usage and runtime of the experiments, allowing this study to include more experiments and more nodes in each experiment, while at the same time providing a sufficient dimensionality to produce high quality embeddings.

- The number of **negative samples** per batch is 64. This number may seem low in compared to the size of a batch, but previous researches have shown that when the input data is large, a smaller amount of negative samples is sufficient to achieve good results.

- **Vocabulary size** is 200.000 unique words. It provides a good enough

representation of the most relevant words in the corpora without skipping specific terms. This value modifies the size of the weight matrices increasing the number of rows.

- **Learning rate** is fixed at 0.1.

The former set of parameters is shared by all executions in the following experiments, independently of executing Word2Vec or Federated Word2Vec. But specifically for Federated Word2Vec, the number of organizations to simultaneously collaborate is an extra parameter needed to simulate a real scenario. It is always fixed to 10 organizations because it is the maximum number that can be created, limited by the available resources in in the machine used for this study.

4.2 Proving convergence of Federated Word2Vec

4.2.1 Objective

This research is focused on answering, as stated in Section 3.1, the following question: "What level of performance does Federated Word2Vec achieve when compared to centralized Word2Vec?"

Therefore, the goal of this first experiment is to provide a proper answer to this question, testing the convergence of Word2Vec with Federated Learning. If the model converges, other evaluation metrics are tested such as similarity, analogy and categorisation.

4.2.2 Setup

Two models are trained in this experiment: Word2Vec and Federated Word2Vec. Both are trained with the same parameters. The dataset contains random articles from Wikipedia with a file size of 254 MB. It is divided into training (229 MB) and validation (25 MB) datasets. For Federated Word2Vec, the datasets were divided into 10 equal partitions, one per external node, for both training and validation datasets.

The models were trained for 5 millions Word2Vec iterations and compared them with 500.000 Federated Word2Vec iterations, because Federated Word2-Vec processes 10 times more data. So this allows us to compare the two approaches after they have seen the same amount of data. However, we also test 2 millions vs 5 millions, as this instead would allow us to understand whether Federated Word2Vec allows us to scale out the training and achieve much better results in similar wall-clock time. We did not use 5 millions iterations for Federated Word2Vec to reduce the experiments execution time, because it would have meant to duplicate the training time, from 4 days to 8 days, reducing time to make other experiments.

Later, the same experiment is run but with a larger dataset for Federated Word2Vec. Its size is 1,74 GB, divided into training (1,6 GB) and validation (140 MB) datasets. Following the same process, the datasets were divided into 10 equal partitions, one per external node.

Another point of comparison is the number of words in each dataset. While the initial dataset has 8.776.470 words in total and 291.012 unique words, the larger dataset has 545.937.033 words in total and 1.556.883 unique words, such a large number of unique words is because there are words like numbers or plurals. It is important to take into account the huge difference in size when analyzing the results in the next section.

4.2.3 Results

Small dataset

Validation loss is the metric selected to perform the analysis of convergence of Federated Word2Vec and a comparison with Word2Vec. In Figure 4.1, there are two graphs representing the validation loss of training with the same dataset, on the left the baseline model and on the right Federated Word2Vec. In order to compare the two models when both have processed the same amount of data, Federated Word2Vec is cut in epoch 70, which is the iteration 500.000.

The loss of Word2Vec presents around a value of 10^4. It is stable, but with a small descending trend. On the other hand, although Federated Word2Vec does not reach the same loss (it is 10 times greater) its trend is clearly decreasing. To check if the trend continues, Figure 4.2 illustrates the validation

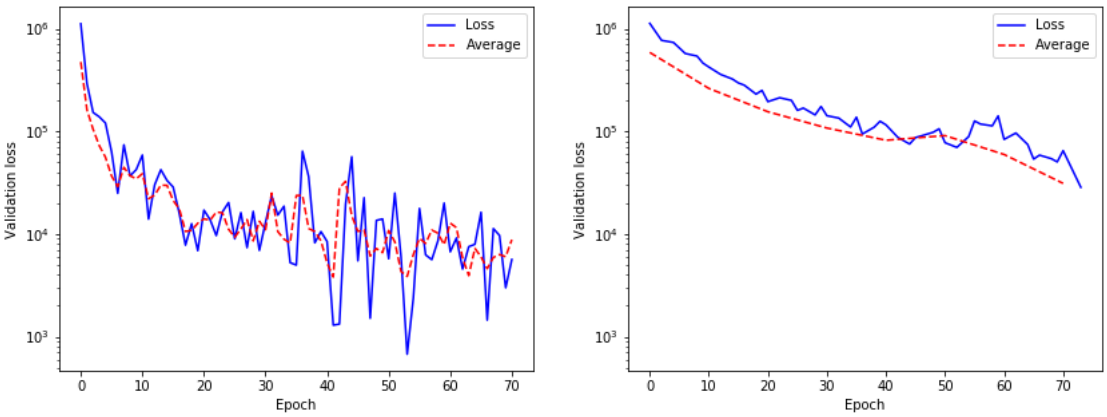


Figure 4.1 – On the left, validation loss per epoch in a full execution of Word2Vec. On the right, validation loss per epoch of the first 500.000 iterarions of Federated Word2Vec. The red lanes represent the average of the validation loss calculated by aggregating all previous values from each epoch. The Y-axis is in logarithmic scale.

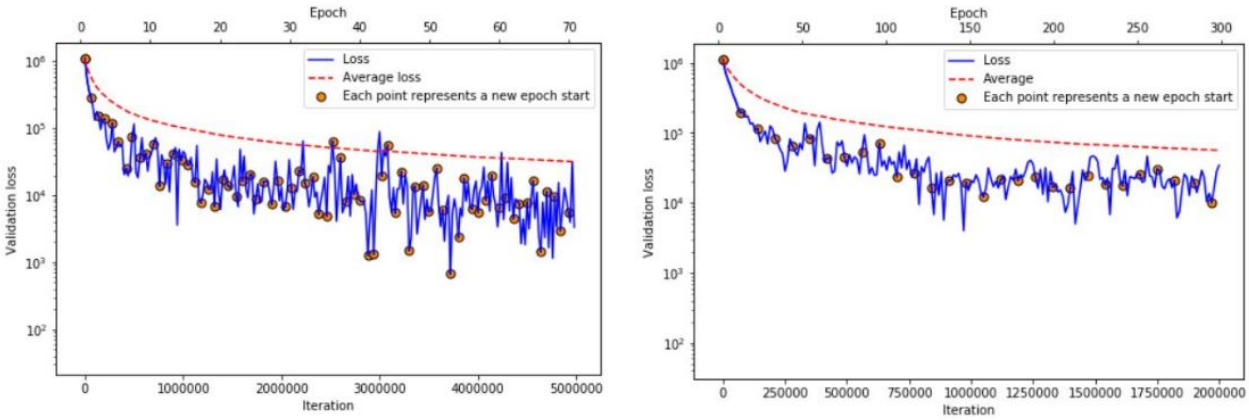


Figure 4.2 – On the left, validation loss per iteration in a full execution of Word2Vec. On the right, validation loss per iterarions in a full execution of Federated Word2Vec. The red lanes represent the average of the validation loss calculated by aggregating all previous values from each epoch. The Y-axis is in logarithmic scale.

loss in terms of iteration for the full execution, 2 millions of iterations. The loss keeps going down until 1 million iterations when it stabilises.

Overall, the two models provide very similar results, the only difference

being a small advantage of Word2Vec in the early phases of training. Word2Vec is faster at the beginning as it can be seen in the slope of the average loss of the first 10 epochs. The loss has a similar behaviour for both models, although Word2Vec seems a bit better. Thus, Federated Word2Vec presents similar characteristics as Word2Vec, but with little differences.

Large dataset

The training of Federated Word2Vec with a large dataset presents improved results compared to the previous graphs. Figure 4.3 freezes the training in the iteration 500.000 as it was done in Figure 4.1. The number of epochs is fewer than in the former experiment but the loss presents a clear downward trend with a steeper slope.

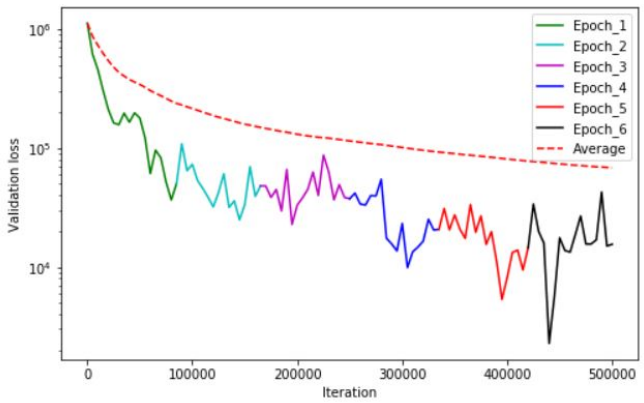


Figure 4.3 – Validation loss of the first 500.000 iterations of Federated Word2Vec with a larger dataset, divided by epoch. The red lanes represent the average of the validation loss calculated by aggregating all previous values from each epoch. The Y-axis is in logarithmic scale.

In Figure 4.4, where the execution continues until iteration 2 millions, the loss keeps decreasing reaching values of 10^3, something that did not happen in the baseline graph in Figure 4.2.

Consequently, Federated Word2Vec seems to work better with larger datasets as it benefits from learning from multiple sources at the same time. The results show that Federated Word2Vec is not better, and might perform slightly worse, than Word2Vec under the same settings. However, it is proven

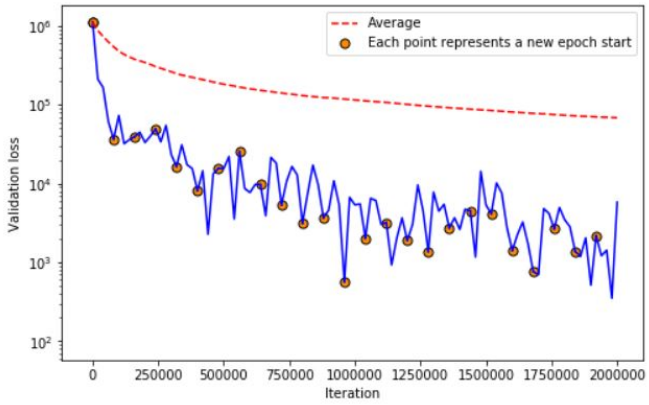


Figure 4.4 – Validation loss of a full execution of Federated Word2Vec with a larger dataset, represented in blue. The red lanes represent the average of the validation loss calculated by aggregating all previous values from each epoch. The Y-axis is in logarithmic scale.

that Federated Word2Vec has a similar convergence pattern to Word2Vec and easily scales to a large dataset.

Similarity, analogy and categorisation

Once proven that Federated Word2Vec converges, it is important to measure the quality of the word embeddings that it generates. In the following Tables 4.1, 4.2 and 4.3, word embeddings are tested against a batch of several similarity, analogy and categorisation tasks, mentioned in Section 3.4. There are three models to compare: Word2Vec (**W2V**), working as the baseline, Federated Word2Vec trained with a short dataset (**Fed W2V**), and Federated Word2Vec trained with a large dataset (**Fed W2V large**).

Overall, it is important to notice that the values on the tables are correlations, hence the closest the value is to 1 the most similar the ranking got by the model is to the human ranking. Most of the values, in particular for analogy tasks, are almost 0, meaning that there is no correlation between both rankings. It is an evidence for **W2V** baseline and **Fed W2V** where only the values start to be separated from 0, especially for categorisation tests.

Fed W2V large outperforms the rest of the models in almost all tests but 2 out of 3 analogy tests. There is a remarkable difference between the correlation

	MEN	WS353	WS353R	WS353S
Fed W2V large	**0.13496**	**0.13570**	**0.22111**	**0.10276**
Fed W2V	0.050589	−0.03028	0.08934	−0.08783
W2V	0.044729	−0.01713	0.10686	−0.07746
	SimLex999	**RW**	**RG65**	**MTurk**
Fed W2V large	**−0.06464**	−0.106	**0.01527**	**0.25796**
Fed W2V	−0.08652	**−0.06096**	−0.04742	0.19638
W2V	−0.07369	−0.08612	0.00852	0.23330

Table 4.1 – Similarity metrics for Word2Vec, Federated Word2Vec and Federated Word2Vec with a larger dataset. Values are correlations, and thus vary in the range [-1,1] with 1 being the best.

	Google	MSR	SemEval_2012
Fed W2V large	**0.07286**	0.052	0.02239
Fed W2V	0.02404	**0.0675**	**0.05293**
W2V	0.01156	0.019375	0.048072

Table 4.2 – Analogy metrics for Word2Vec, Federated Word2Vec and Federated Word2Vec with a larger dataset. Values are correlations, and thus vary in the range [-1,1] with 1 being the best.

	AP	BLESS	Battig	ESSLI_2c	ESSLI_2b	ESSLI_1a
Fed W2V large	**0.22**	**0.26**	**0.16**	**0.36**	**0.45**	**0.43**
Fed W2V	0.16	0.25	0.10	0.31	0.42	0.39
W2V	0.17	0.23	0.11	0.33	**0.45**	0.41

Table 4.3 – Categorisation metrics for Word2Vec, Federated Word2Vec and Federated Word2Vec with a larger dataset. Values are correlations, and thus vary in the range [-1,1] with 1 being the best.

achieved by **Fed W2V large** and the other models. Despite the fact that the model is still far from the metrics obtained by the best models*, it provides better quality embeddings, even although its loss function provided similar results.

It is interesting to look at the difference between **W2V** and **Fed W2V**, as these two have the same amount of data. They both do not perform well, but it is not clear if, at least, one is better than the other. So, all the metrics for

* To check the table of the best results available in this benchmark, see the following website https://github.com/kudkudak/word-embeddings-benchmarks/wiki.

each benchmark were summed up to compare, overall, their performance. **Fed W2V** has a total score of $1,798$ and **W2V** has a total score of $1,918$. The maximum score that a model can obtain is 17, thus, the score in percentage obtained by the models are $10,57$ and $11,28$, respectively. There is no visible difference between the models. It is the expected result because both models have processed the same amount of data from the same dataset. Moreover, Federated Word2Vec does not seem to harm the training, not presentig a big gap of performance against Word2Vec.

Word embeddings in 3D space

Figure 4.5 illustrates a 3D representation of the word embeddings from the two first models trained with the same short dataset. A PCA dimensionality reduction is applied to produce the visualisation. From the figure, it can be noticed that the vast majority of words are concentrated around the same point, while a group of words spread out of the core following an *L* pattern.

The further the point is from the agglomeration of points, the better the word is learnt. In other words, if a word is in the tail of the *L*, the model captured its meaning better in comparison to the rest of the words. To prove it, the right side of the figure indicates the position of a set of words with the linguistic root *house*. The root is better represented than other morphological constructions from it such as the plural *houses* or the derived noun *household*. This happens because it is more likely that the root appears in a text than its derived words. There are more complex tokenization approaches like WordPiece that can identify sub-concepts in complex words, in order to allow the system to learn them better. For example, the word greenhouse would be replaced by the token *green* and the token *house*. Complex contextual embedding models like BERT can take advantage of this, as they can use attention layers to understand the juxtaposition of the two words and can reuse the separate knowledge of the two words when computing their contextualized embeddings. However, Word2Vec would not be able to take advantage of this tokenization approach.

Comparing Word2Vec, in 4.5a, and Federated Word2Vec, in 4.5b, the tail is more populated for the second model. It encapsulates better the context of more words. Some derived words are also closer to the end of the tail. The size of the dataset does not affect the result because both model were trained

with the same data. The other unique difference in the setup is the number of iterations. Federated Word2Vec was trained during 2 millions iterations while Word2Vec trained during 5 millions. Thus, Federated Word2Vec processed 4 times more data. It gave more time to the model to acquire more knowledge from less common words.

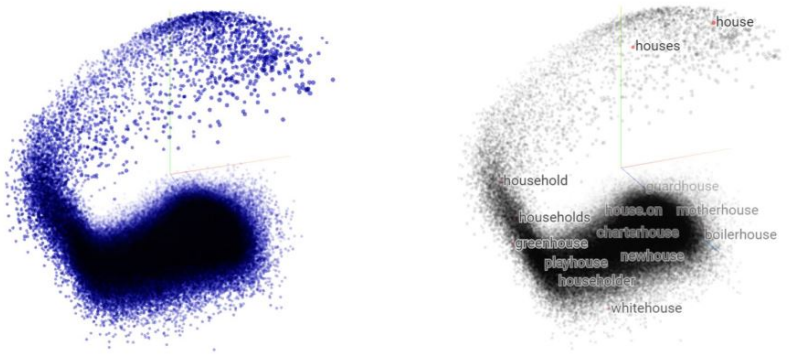
Another perspective to analyse the Figure is the spreading of the tail. The distance between words is larger in 4.5a than in 4.5b. Word2Vec managed to separate the words more from each other, which would count as a better results, but it is only among very common words. Federated Word2Vec managed to extract from the core of unknown words more terms, but it would need more time to separate them from each other. Thus, it is not possible to conclude which model is better in terms of the distance between words in the tail. It is clear that the models are learning in different ways. But it is possible to theorise about hypothesis that can fit here to explain this behaviour. First, one hypothesis could be that the extraction of words from the core is something that each node can do individually. So, the 10 nodes together manage to focus on different sets of words and thus, move more words out of the core. On the other hand, another hypothesis is that correctly and clearly positioning and spreading the extracted words from each other requires a lot of coordination. It happens more slowly in a distributed setup. However, we do not have any way to verify this and so more research is needed on this aspect. In general, the topic of how a model learns internally is still quite unexplored.

4.3 Federated Word2Vec with categorised data

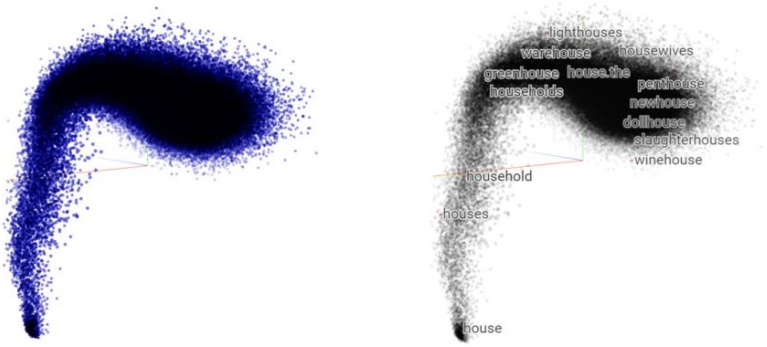
4.3.1 Objective

Once it is proven that the model works, the next experiment is focused on simulating a real scenario as the one represented in Figure 3.3 in Subsection 3.1.3. In this scenario, there are 10 organizations that want to collaborate to create a good word representation of the archives stored in their databases. The topic of the archives varies depending on the organization, there are only 5 different topics spread into the organizations. Thus, in this simulation, every topic is represented in the data of two organizations.

This experiment wants to verify whether sharing the knowledge of each



(a)



(b)

Figure 4.5 – 3D representation of word embeddings of Word2Vec in (a) and Federated Word2Vec in (b). On the left side, the shape of word embeddings without any modification. On the right side, the same representation but applying a search based on the linguistic root *house*.

.

specific topic to the central model produces global embeddings that provide a universally acceptable representation of each word while still preserving topic-specific meaning.

4.3.2 Setup

The topics of the datasets, mentioned in 3.3.1, used in this experiment are five: biology, finance, geography, history and sports. The experiment consists

of one baseline Word2Vec model trained with only the finance category, while the second model is Federated Word2Vec trained with 10 external nodes and 5 topics.

The size of the datasets is 278 MB for each category, divided into training (151 MB) and validation (27 MB) sets. All categories have the same size. So the total data processed per epoch is 1.51 GB.

Word2Vec is trained for 5 millions iterations while the execution in the case of Federated Word2Vec is stopped when 2 millions of iterations are reached. It follows the same scheme as in the previous experiment.

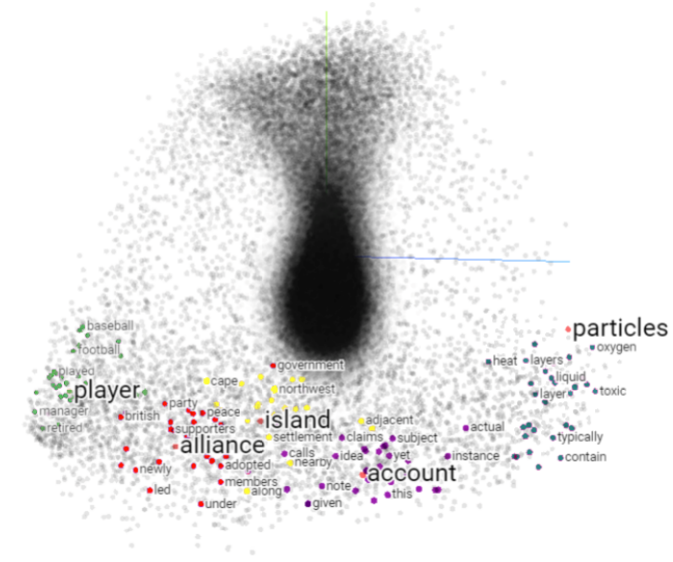
4.3.3 Results

For this experiment, the focus is on the visual representation of the embeddings. The aim is to validate if the collaborative training maintains clusters of words per category. The cosine distance is chosen as the distance to calculate the clusters. Then, the clusters identified in Word2Vec trained in one category are compared with the ones present in Federated Word2Vec trained on all 5 categories, in order to understand the impact of a topic-specific dataset during the training. Once the topic-specific datasets were prepared, for each topic you identify a word that you will use to pinpoint the position of that topic in the embedding space. Then, these positions are checked to see how they differ between models trained under different conditions. In addition, between both groups, some words are the same to perform a proper comparison.

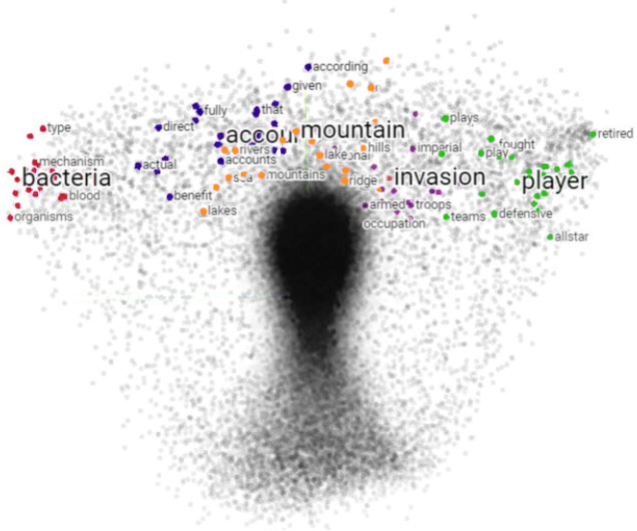
The clusters shown in Figure 4.6 for two different set of words 4.6a and 4.6b represent the 25 and 35 closest neighbours for each word of the set based on the cosine distance, respectively. In both graphs, two communities are delineated for *sports* and *biology*, the cluster is still defined in the same position even changing the word in the second category. The boundaries between the other 3 clusters are not as well defined as the former set of categories. However, although they clash with some points, the communities keep their same spot, even changing the words for *biology* (*bacteria* instead of *particles*), *geography* (*mountain* instead of *island*) and *history* (*invasion* instead of *alliance*). Thus, no matter which word is used as the center of the cluster to identify a category, the relative and absolute positions of the categories will not change much, meaning that the model is doing a good job identifying these categories.

Figure 4.6 clearly shows that the words around a certain topic are organised in groups, picking a particular spot that may depend on the distance of meaning or coincidence of same words between categories.

Digging into the organisation of the word embeddings in communities in the space, Figure 4.7 plots the neighbours of *account*, *bacteria* and *mountain* in a execution of Word2Vec with only *finance* dataset and Federated Wor2Vec with all five categories. The most striking finding is that *account* present meaningless words such as *{still, to, their}* in its community when trained with only *finance* dataset, while the execution of Federated Word2Vec shows more specific context words such as *{accounts, benefit, personal}*.



(a)



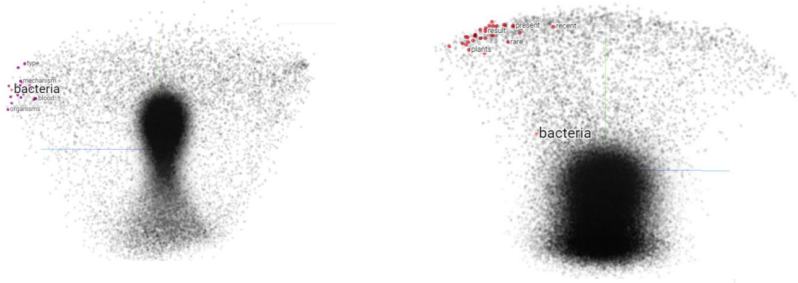
(b)

Figure 4.6 – 3D representation of word embeddings of Federated Word2Vec for the set of words *player, alliance, island, account, particles* (a) *player, invasion, mountain, account, bacteria* in (b). The coloured points are the 25 closest neighbours for set (a) and 35 for set (b).

.

For the other two sample words, the trend is similar. *Bacteria* and *mountain* present in their neighbourhoods meaningful words when trained with Federated Word2Vec, for example, *{blood, organism}* and *{lake, hills, rivers}*, respectively. And more generic words in the execution of Word2Vec *{plants, result, rare}* for *bacteria* and *{mount, highest, southern}* for *mountain*.

(a)



(b)

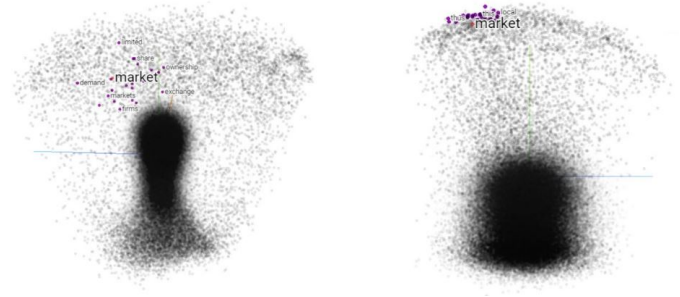
(c)

Figure 4.7 – 3D representation of word embeddings of Federated Word2Vec with five categories on the left and Word2Vec with only *finance* category on the right. The words are *account* (a), *mountain* (b) and *bacteria* (c). The coloured points are the 25 closest neighbours for each words.

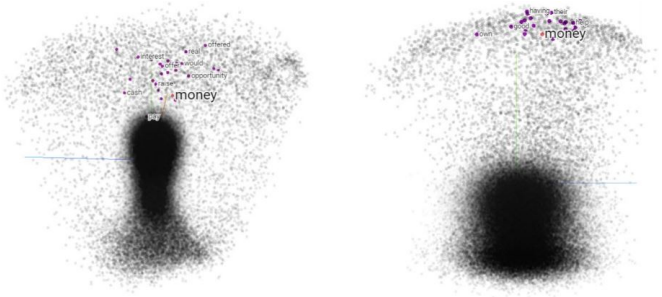
.

This behaviour is explained because those two words belong to a category which is not directly trained in that execution. But, the fact that certain categories do not give good results in Word2Vec is expected because it did not process data from them. However, the bad results for the word *account* are not expected. Joining both findings will confirm the following hypothesis: *it is important for organizations to cooperate, as cooperation provides models that are not only globally good, but also locally better than locally-trained models.*

In order to confirm the former hypothesis, two more generic words from the finance topic are studied: *market* and *money*. Figure 4.8 adds more contrast to the results and allows to discard that *account* is an outlier. The new results confirm the hypothesis. The resultant neighbourhood of the word *money* when Word2Vec is trained only with the financial dataset, it presents generic words such as *good, help, having*. The same behaviour is also found for the word *market* under the same training setup, a subset of neighbours is *thus, this, local*. For Federated Word2Vec, trained with 5 categories, the neighbours obtained are different. They gather meaning from the finance topic for both words. In the case of *money*, a subset of the neighbours is *interest, cash, offer*, and for *market*, the subset is *share, demand, exchange*.



(a)



(b)

Figure 4.8 – 3D representation of word embeddings of Federated Word2Vec with five categories on the left and Word2Vec with only *finance* category on the right. The words are *market* (a) and *money* (b). The coloured points are the 25 closest neighbours for each words.

.

Loss per node

Figure 4.9 shows training loss for each node trained with a unique category. Each category is trained by two nodes, half of the nodes are not shown because the data collected is similar. All graphics present the same trend, only changing the initial values. Hence, the convergence is not affected by the topic of the text. It is explained by the proportion of number of occurrences of category related words and common words. Although a text is specialised in a topic, the proportion between common and specific words is still in favour of common words appearances.

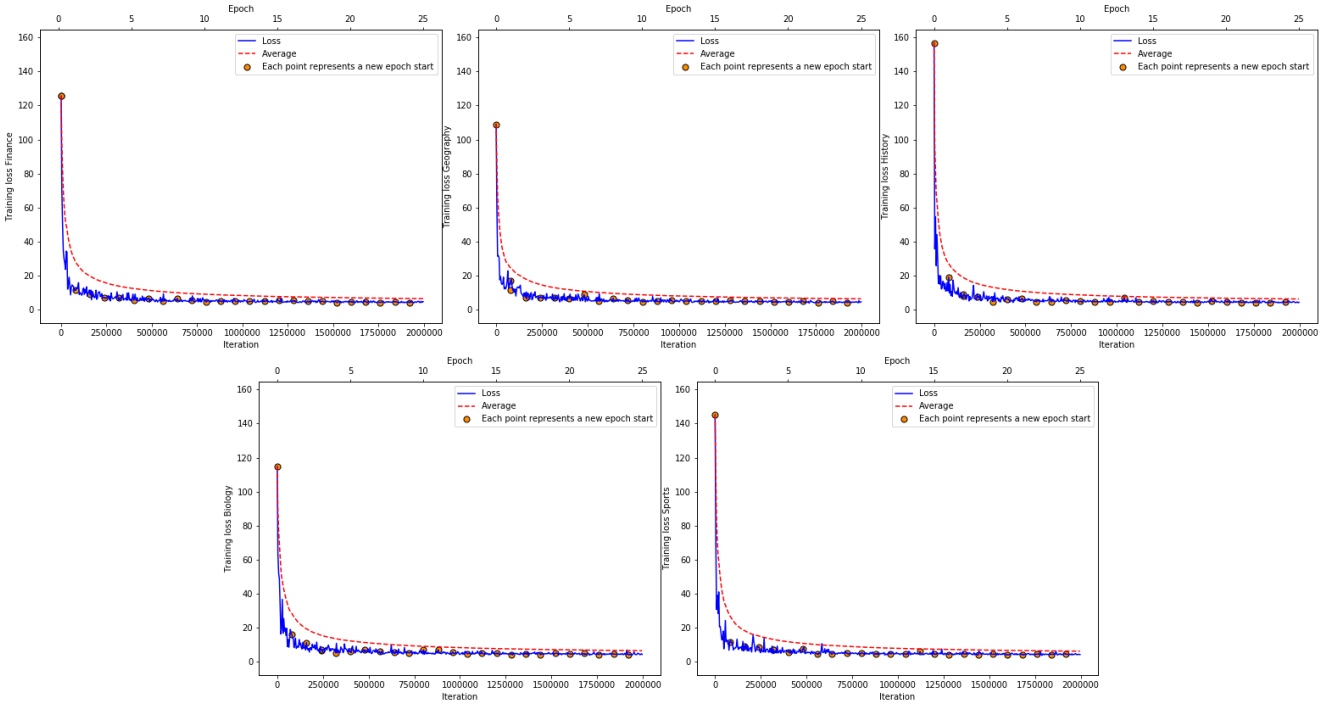


Figure 4.9 – Training loss, represented in blue, of a full execution of Federated Word2Vec with five categories, one per graph. The red lanes represent the average of the training loss calculated by aggregating all previous values from each epoch.

4.3.4 Similarity, analogy and categorisation

Again, the quality of word embeddings is an important evaluation metric to compare both models obtained from Word2Vec trained only in one category and Federated Word2Vec trained in 5 categories. Similarity, analogy and categorisation tasks are the same as in the previous experiment.

The results follow the same trend recognised in Tables 4.4, 4.5 and 4.6, when the same models were trained with scatter data. Federated Word2Vec scores higher in almost every rank. The scores are also similar, it means that the topic of the data does not affect the scores. However, it is a fact that the larger the data, the better the scores.

	SimLex999	RW	RG65	MTurk
Fed W2V	**0.13496**	**0.13570**	**0.22111**	**0.10276**
W2V	0.11127	−0.00092	0.10664	−0.04424
	MEN	**WS353**	**WS353R**	**WS353S**
Fed W2V	**−0.06464**	**−0.106**	**0.01527**	**0.25796**
W2V	−0.06752	−0.09626	−0.02272	0.21584

Table 4.4 – Similarity metrics for Word2Vec model trained in one category and Federated Word2Vec trained in 5 different categories, including the one used to train Word2Vec. Values are correlations, and thus vary in the range [-1,1] with 1 being the best.

	Google	MSR	SemEval_2012
Fed W2V	**0.07286**	**0.052**	0.02229
W2V	0.01294	0.00962	**0.04411**

Table 4.5 – Analogy metrics for Word2Vec model trained in one category and Federated Word2Vec trained in 5 different categories, including the one used to train Word2Vec. Values are correlations, and thus vary in the range [-1,1] with 1 being the best.

	AP	BLESS	Battig	ESSLI_2c	ESSLI_2b	ESSLI_1a
Fed W2V	**0.22**	**0.27**	**0.16**	**0.36**	**0.45**	0.43
W2V	0.17	0.23	0.12	0.33	**0.45**	**0.48**

Table 4.6 – Categorisation metrics for Word2Vec model trained in one category and Federated Word2Vec trained in 5 different categories, including the one used to train Word2Vec. Values are correlations, and thus vary in the range [-1,1] with 1 being the best.

4.4 Federated Word2Vec with unbalanced data

4.4.1 Objective

This experiment is a natural continuation of the experiment in Section 4.3. The idea is to go deeper in the analysis of the structure of the word embeddings generated when many organizations collaborate together. Here, the experiment follow the same structure designed for the previous one: 10 organizations collaborating, 5 different topics spread into those organizations per pairs. The difference is that the datasets vary in size.

This scenario wants to prove if the situation analysed in the previous section is kept with different sizes of data in the organizations.

4.4.2 Setup

The topics of the datasets are exactly the same as in the previous experiment; biology, finance, geography, history and sports. A Federated Word2Vec model is trained with 10 external nodes and 5 topic related datasets with different sizes. This model is compared to the results of Federated Word2Vec trained with datasets of same size.

The size of the datasets are: finance 74 MB, geography 434 MB, history 539 MB, biology 314 MB and sports 342 MB. The size in total of the data processed is 1.7 GB. The validation datasets is a collection of a 10% of the size of the original datasets for each topic. The total number of iterations is 2 millions.

4.4.3 Results

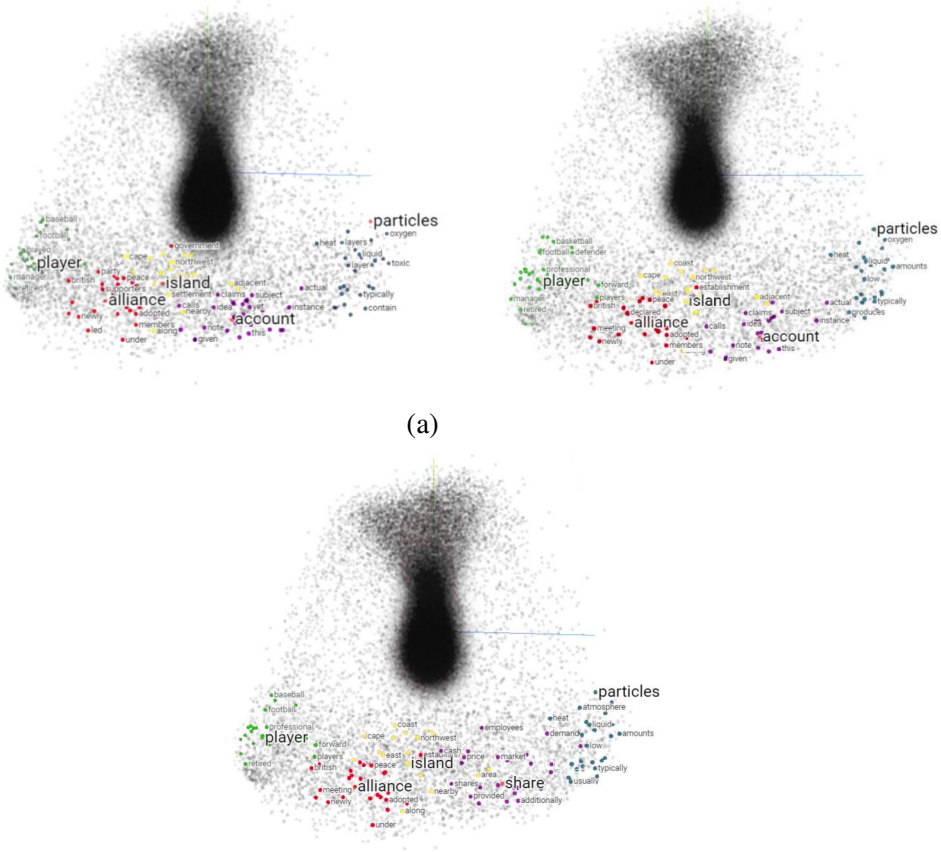
The visualisation of the word embeddings for the unbalanced size of datasets has similar results to the plots with same size datasets. It can be seen in Figure 4.10a that the spot in the space of the clusters shaped for each category is still the same. The only noticeble change is that the shorter dataset, finance category, is a bit more isolated and does not overlap with the history and geography clusters as much. This situation could be happening because the model is not sure about where the finance cluster should be, being wider than in the previous experiment. Moreover, the finance dataset is smaller so there are less chances for words from finance category to interact with other words, ending up more isolated. In addition, from the first experiment we know that a word is placed in the tail of the distribution if the model has capture the meaning of it. It is facilitate because the word was processed more times by the model because the dataset is smaller.

This result provides different possible behaviours of the model. On one side, if the goal is to learn a perfect representation of a topic, while keeping base knowledge from other words, this behaviour could be good. However, even though this situation could be, at first sight, the objective, it should not

be the desired behaviour. It would mean that the model is learning very well what words belong to finance, but it does not have a chance to learn how finance topic as a whole interacts with other topics, such as geography or history. This is a way of learning that it is not interesting from a general collaborative perspective. Therefore, the expected behaviour should be a model that learns decently topic-specific words, even if the dataset is short, but gathers information about the relations with other topics.

In order to clarify the results, another word is selected to analyse the situation. Figure 4.10b illustrates that the cluster of the word *share* is placed in the same spot as the cluster of *account* but it is more spread. This is a distribution than could be foreseen to obtain in a training with datasets of same size. Therefore, the model presents the correct performance, achieving the better interaction among topics, while preserving information about the topic from the short dataset.

In general, there is no huge impact in the organisation and quality of the embeddings. Hence, organizations with shorter corpora can still collaborate with larger organizations.



(a)

(b)

Figure 4.10 – In (a), a 3D representation of word embeddings of Federated Word2Vec with five categories on the left with equal size of datasets and the same model trained with same categories but with different sizes of datasets on the right. The words are *account, alliance, island, particles and player*. In (b), same model trained with different sizes of datasets but including a different word *share* for the finance category. The coloured points are the 25 closest neighbours for each words.

.

Loss per node

Figure 4.11 shows training loss for each node trained with a unique category. Each category is trained by two nodes, half of the nodes are not shown because the data collected is similar. Even though the dataset used to train each node is

different, all plots present the same trend, converging in a similar point before the iteration 200.000. It does not mean that the training should be stopped at that point because the embeddings weight matrix keep changing, assimilating the context of the words and making new relations among them.

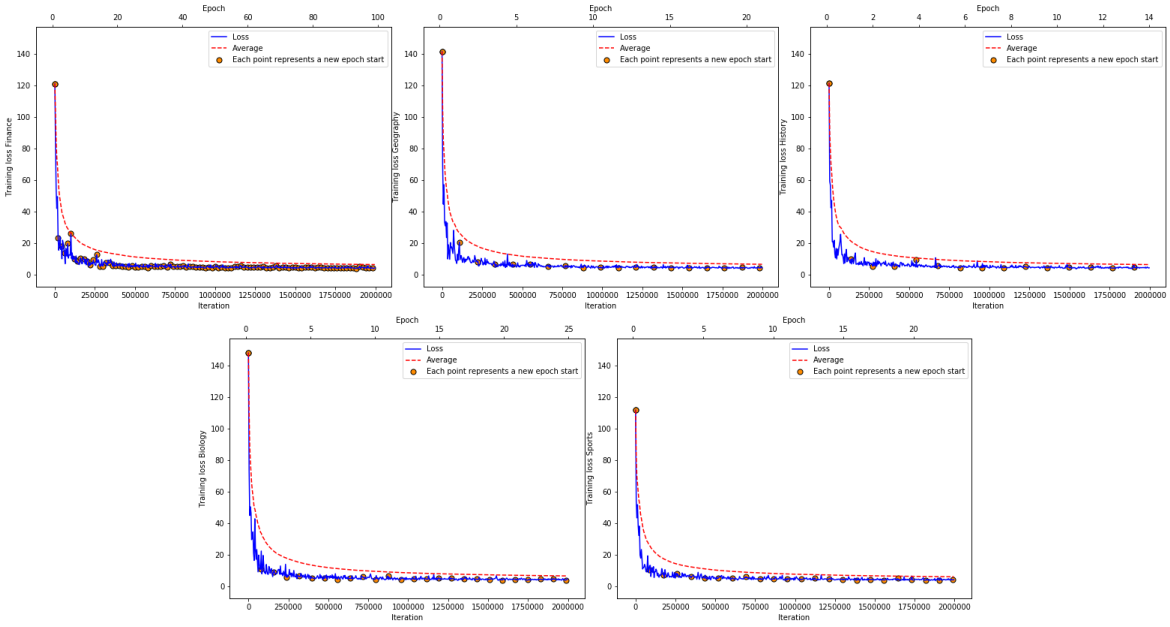


Figure 4.11 – Training loss, represented in blue, of a full execution of Federated Word2Vec with five categories, one per graph. The red lanes represent the average of the training loss calculated by aggregating all previous values from each epoch.

The convergence is not affected by the size of the dataset. The number of occurrences of a word is still the predominant factor to obtain a high-quality vector representation of a word.

4.5 Hardware requirements

This section addresses the hardware requirements to train correctly a Federated Word2Vec model. From the computational power perspective, the computational load is distributed among all nodes. It means that training Federated

Word2Vec with 10 external nodes will take the same amount of time as training it with 5 external nodes, under the assumption that the amount of data is fixed in all nodes. Each node trains one Word2Vec model calculating the gradient, but not updating the weights because it is the task of the central node. The amount of time required to train Word2Vec in a machine, with the specifications mentioned in Section 3.3, was $16,96$ hours for 5 millions of iterations.

4.5.1 Network bandwidth

The traffic over the network can be sometimes a bottleneck in a distributed architecture. In the case of Federated Word2Vec, it is still a concern to take into account. Each node transfers the matrix of gradients to the central node per iteration. The amount of bytes transferred depend on the number of rows n, vocabulary size, and columns m, embedding size, of the embedding matrix and the NCE weights matrix. It is a total of $n \times m$ bytes, and each position (n_i, m_j) of the matrix in a numpy array has a size of 4 Bytes.

It is important to pay attention to how the gradients can be compressed for efficient transfer. There is a lot of research going on in this field where different approaches are being explored. One alternative to sending the whole matrix is to just send the non-zero entries along with their positions in the matrix, This can be even more effective by using gradient sparsification[65], which consists of rounding small values to zero, further reducing the number of entries to transfer. An orthogonal line or research consists in reducing the number of bits required to send each value. Finally, these techniques can be combined to achieve a much higher compression rate[66].

However, in this particular implementation, there is an option in TensorFlow to define the gradient with indexes, similar to the first approach of gradient compression explained before. The direct application of compression techniques are one of the future work directions of this research. With this TensorFlow option, only the rows modified in the batch are saved in the memory. The gradient is represented by a list of indexes to access the rows in the weight matrices, and a matrix with only the gradients of the rows to be updated. It requires less memory space, hence less information to share with the central node, reducing the network load.

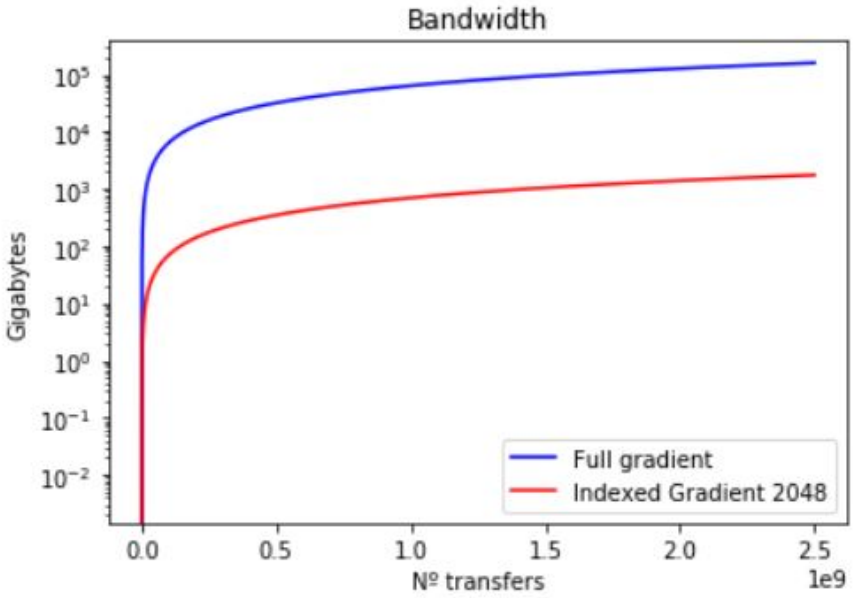


Figure 4.12 – Number of bytes transferred though the network from a node to the central node sharing whole gradient matrix, in blue, and using the indexed gradient from TensorFlow, in red. Bytes are aggregated per iteration. The Y-axis is in logarithmic scale.

Figure 4.12 shows the total aggregated amount of bytes transferred in function of the number of iterations for both alternatives explained before. It is clear that the indexed gradient is a better approach to reduce the network load without compromising the training performance. However, the batch size affects the size of the gradient share, so bigger batch sizes will produce bigger gradients, but still smaller than the full gradient matrix.

Chapter 5

Discussion

5.1 Benefits

This research encourages the use of massively distributed protocols, in particular, Federated Learning. It can mitigate many of the systemic privacy risks of most current machine learning applications, complying with sensitive private data from users, and costs, reducing the expenses in powerful machines, resulting from traditional, centralised Machine Learning and Data Science approaches.

This study expands the knowledge and capabilities of Federated Learning in the area of NLP and in particular of Word2Vec. It opens research questions to continue the work in the future, keeping the explosive growth in the trend of Federated Learning.

5.1.1 Ethical considerations

In recent years, governments and organizations have introduced several new data regulations to protect the privacy of sensitive information from users. This is an answer to the huge amount of data that users shared with organizations in the Internet through their smart devices. An example is the recent implementation of the General Data Protection Regulation (GDPR) issued by the European Union. Protocols like Federated Learning provide a suitable and successful

approach for Machine Learning models in order to comply with those new regulations. Although this research does not aim to provide a security measure to solve possible attacks to the protocol that may occur, leading to data leaks. It creates an awareness of the need to pay attention in that matter in future works when working in massively distributed or decentralised protocols.

5.2 Limitations

Massively distributed Machine Learning field is large and recently discovered, that has opened a wide area of research with many unknown scenarios yet to explore. This research stated a purpose in Section 1.3 that address only a portion of these many important scenarios. Thus, the results provided in this study are only valid when the assumptions, resources and conditions that were explicitly considered do not differ from the experimental setup.

It was stated in Section 1.2 the problem this research wants to address. It simulates a real scenario where many variables such as asynchronous communication, system fails or different execution times from a wide variety of devices are not considered. All these circumstances may have a huge impact in the behaviour of the protocol, varying the results obtained in this research. Hence, many interesting scenarios were not modelled or tested due to the limitations of the resources. While this research had access to a quite powerful machine, state-of-the-art NLP research requires amounts of data and computational resources on a much larger scale. The resources available were enough to locally simulate a small number of external nodes, but a large-scale simulation was not possible, and thus this work is not representative of all potential real-world scenarios.

Furthermore, only one word embedding model was tested. It provides a minimum measure to ensure the meaningfulness of the results because next generation of models are also based on Artificial Neural Networks, but with improved architecture designs.

Privacy and security were two concepts mentioned in this research as one of the main reasons to encourage the development of massively distributed protocols. However, no privacy or security considerations were taking into account in the experiments. Although they are a core part of the protocols, it was not the purpose of this study to include them. But they should always be

taken into consideration in a real case scenario.

5.3 Future work

Massively distributed protocols like Federated Learning are experiencing an explosion in the number of studies conducted in the field. This research adds light in the viability of the protocol with new sorts of models like Word2Vec, investigating the benefits of using Federated Learning in the NLP area.

But there are still open paths to continue the research in this direction. From one side, it is possible to expand the knowledge testing larger and more comples models such as ELMo and BERT. It will provide better results, closer to the state of the art benchmarks. Moreover, it would be an option to still work with Word2Vec but exploring other approaches of Federated Learning, for example, Federated Averaging protocol, in order to reduce even more the information shared through the network.

As mentioned in Section 4.5.1, Federated Learning protocol is based on the transfer of the matrix of gradients. It creates a dependency with the bandwidth of the network connection of the devices that are cooperating in the training because the gradient is sent through it. Thus, gradient compression techniques are a must to add in the future. Novel techniques are appearing in order to find the better compression, so there is room to test different techniques to improve the behaviour of Federated Learning, in particular with NLP models because the non-zero gradients reveal exactly which words the user has entered on the device if the features are a sparse bag-of-words. It would provoke a leak of information if a malicious software has access to the gradients.

Furthermore, Federated Learning provides the possibility of collaborative training. It can enable new techniques for partial personalisation of the trained representations for each organization that participates in the process, resulting in better quality and data-specific models.

5.4 Conclusions

The union of all key factors: privacy concerns, huge amount of accessible data and many available smart devices; leads to a solid growth of the massively distributed protocols. Optimistically, new protocols and techniques will appear in the field, improving current state of the art. Together with Machine Learning algorithms, more powerful and safer application will be able to be developed, reaching a new peak of performance.

Following this path, this research wants to add new capabilities of the Federated Learning protocol in the field of NLP. The purpose of this study is to implement and test the viability of a distributed, efficient, data-private approach that allows a small number of organizations, each owning a large private text corpus, to train global word representations. The results could be applicable to real scenarios of collaborative training. The main contributions of this work are:

- The viability of training NLP models like Word2Vec under the Federated Learning protocol. The result is what is called Federated Word2Vec. The convergence times are, at least, at the same level of the widely tested Word2Vec. In addition, Federated Word2Vec benefits from training with large volumes of data, processing data faster from different sources. It results in better word representations, getting better scores in similarity, analogy and categorization benchmarks than Word2Vec, under the same training setup.

- The importance for organizations to cooperate, as cooperation provides models that are not only globally good, but also locally better than locally-trained models.

- The quality of vector representations is not affected by the size of the corpora of the organizations. It facilitate the cooperation among small and large organizations.

Therefore, the answer, and the main conclusion of this thesis, to the research question introduced in Section 1.3 is the following:

Yes, it is viable to obtain a high-quality word vector representation using a massively-distributed protocol like Federated Learning without compromising

execution time, performance and resources. The importance of cooperating among different organizations is also proven as the resultant vector representation obtained present similar or better quality.

References

[1] M. Minsky and S. Papert, *Perceptrons: an introduction to computational geometry*, 1969.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," May 2015. [Online]. Available: https://www.nature.com/articles/nature14539

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016.

[4] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013. doi: 10.5555/2999792.2999959

[5] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[7] D. Liu and T. Miller, "Federated pretraining and fine tuning of bert using clinical notes from multiple silos," 2020.

[8] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, Inc., March 2017.

[9] D. Bzdok, N. Altman, and M. Krzywinski, ""statistics versus machine learning"," vol. Nature Methods 15 (4), March 2018. doi: 10.1038/nmeth.4642. PMC 6082636. PMID 30100822

[10] A. Gulli and S. Pal, *"Deep Learning with Keras"*. Packt Publishing, April 2017.

[11] Y. Zhou, D. Wilkinson, R. S. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," June 2008. doi: 10.1007/978-3-540-68880-832

[12] T. K. Ho, "Random decision forests," vol. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, August 1995. [Online]. Available: https://web.archive.org/web/20160417030218/http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf

[13] C. Cortes and V. Vapnik, "Support-vector networks," 1995. doi: 10.1007/BF00994018

[14] D. Cires, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," 2012.

[15] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," vol. IEEE Conference on Computer Vision and Pattern Recognition, June 2009. doi: 10.1109/CVPR.2009.5206848

[16] Y. Bengio, R. Ducharme, and P. Vincent., "Imagenet classification with deep convolutional neural networks," *Journal of Machine Learning Research*, vol. 3:1137-1155, 01 2003.

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[19] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," 2012. doi: 10.1109/CVPR.2009.5206848

[20] B. Pang and L. Lee, "Opinion mining and sentiment analysis," vol. Volume 2 Issue 1–2, 2008. doi: 10.1561/1500000001

[21] P. Brown, J. Cocke, S. Pietra, V. Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossin, "A statistical approach to machine translation," vol. Computational linguistics 16, 2, p. 79–85, 1990.

[22] P. Brown, S. Pietra, V. Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossin, "A statistical approach to language translation," vol. n Proceedings of the 12th Conference on Computational Linguistics - Volume 1, 1988.

[23] P. Brown, J. Cocke, S. Pietra, V. Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossin, "The mathematics of statistical machine translation: Parameter estimation," vol. Comput. Linguist. 19, 2, 1993.

[24] H. Ghasemi and M. Hashemian, "A comparative study of google translate translations: An error analysis of english-to-persian and persian-to-english translations," *English Language Teaching*, vol. 9, p. 13, 01 2016. doi: 10.5539/elt.v9n3p13

[25] M. Anusuya and S. Katti, "Speech recognition by machine: A review," vol. Volume 6 Number 3, 2009.

[26] Y. Li, L. Xu, F. Tian, L. Jiang, X. Zhong, and E. Chen, "Word embedding revisited: A new representation learning and explicit matrix factorization perspective," 2015.

[27] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," 2014. [Online]. Available: https://levyomer.files.wordpress.com/2014/09/neural-word-embeddings-as-implicit-matrix-factorization.pdf

[28] A. Osterlund, D. Odling, and M. Sahlgren, "Factorization of latent variables in distributional semantic models," 2015. [Online]. Available: http://www.emnlp2015.org/proceedings/EMNLP/pdf/EMNLP024.pdf

[29] D. Jurafsky and J. H. Martin., *Speech and Language Processing*, October 2019, vol. Chapter 3: N-gram Language Models.

[30] K. S. Jones, *A statistical interpretation of term specificity and its application in retrieval*. Journal of Documentation, October 1972, vol. 28 (1).

[31] S. Rose, D. Engel, N. Cramer, and W. Cowley, *Automatic Keyword Extraction from Individual Documents*, 03 2010, pp. 1 – 20. ISBN 9780470689646

[32] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Proceedings of Workshop at ICLR*, vol. 2013, January 2013.

[33] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," vol. 14, 01 2014. doi: 10.3115/v1/D14-1162 pp. 1532–1543.

[34] T. Shi and Z. Liu, "Linking glove with word2vec," 2014.

[35] Y. Bengio, R. Ducharme, and P. Vincent., "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3:1137-1155, 01 2003.

[36] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *AISTATS'05*, 2005, pp. 246–252.

[37] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. MIT Press and McGraw-Hill, October 2001, vol. Section 16.3.

[38] N. A. Smith and J. Eisner, "Contrastive estimation: Training log-linear models on unlabeled data," 2005.

[39] M. Gutmann and A. Hyvarinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," vol. Volume 9 of JMLR: WCP 9, Appearing in Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy.

[40] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," vol. NIPS'13 2, p. 2265–2273, December 2013.

[41] A. Bondi, "Characteristics of scalability and their impact on performance," 01 2000. doi: 10.1145/350391.350432 pp. 195–203.

[42] A. W. Services, “Introducing amazon ec2 p2 instances, the largest gpu-powered virtual machine in the cloud,” 2016.

[43] J. D. George F Coulouris and T. Kindberg, “Distributed systems: concepts and design,” 2005.

[44] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, “A survey on distributed machine learning,” 2019.

[45] R. Ormándi, I. Hegedűs, and M. Jelasity, “Gossip learning with linear models on fully distributed data,” *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, p. 556–571, May 2012. doi: 10.1002/cpe.2858. [Online]. Available: http://dx.doi.org/10.1002/cpe.2858

[46] A. Soliman, S. Girdzijauskas, M.-R. Bouguelia, S. Pashami, and S. Nowaczyk, “Decentralized and adaptive k-means clustering for non-iid data using hyperloglog counters,” H. W. Lauw, R. C.-W. Wong, A. Ntoulas, E.-P. Lim, S.-K. Ng, and S. J. Pan, Eds. Springer International Publishing, 2020. ISBN 978-3-030-47426-3

[47] L. Giaretta and S. Girdzijauskas, “Gossip learning: Off the beaten path,” in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 1117–1124.

[48] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[49] “Word2vec negative sampling model,” https://medium.com/towardsdatascience/word2vec-negative-sampling-made-easy-7a1a647e07a4, accessed: 2020-05-15.

[50] "Word2vec architecture," https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html, accessed: 2020-05-24.

[51] Y. Yaghoobzadeh and H. Schütze, "Intrinsic subspace evaluation of word embedding representations," 2016.

[52] A. Rogers and A. Drozd, "Intrinsic evaluations of word embeddings: What can we do better?" 08 2016. doi: 10.18653/v1/W16-2507

[53] B. Wang, A. Wang, F. Chen, Y. Wang, and C.-C. J. Kuo, "Evaluating word embedding models: methods and experimental results," *APSIPA Transactions on Signal and Information Processing*, vol. 8, 2019. doi: 10.1017/atsip.2019.12. [Online]. Available: http://dx.doi.org/10.1017/ATSIP.2019.12

[54] E. Bruni, N. K. Tran, and M. Baroni, "Multimodal distributional semantics," *J. Artif. Int. Res.*, vol. 49, no. 1, p. 1–47, Jan. 2014.

[55] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics*, vol. 41, no. 4, pp. 665–695, 2015. doi: 10.1162/COLI_a_00237. [Online]. Available: https://doi.org/10.1162/COLI_a_00237

[56] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, , and E. Ruppin, "Placing search in context: The concept revisited," *In Proceedings of the 10th International Conference on World Wide Web, WWW '01, pages 406–414, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0.* doi: 10.1145/371920.372094. [Online]. Available: http://doi.acm.org/10.1145/371920.372094

[57] T. Luong, R. Socher, and C. Manning, "Better word representations with recursive neural networks for morphology," in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 104–113. [Online]. Available: https://www.aclweb.org/anthology/W13-3512

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

[59] G. Halawi, G. Dror, E. Gabrilovich, and Y. Koren, "Large-scale learning of word relatedness with constraints." [Online]. Available: http://www2.mta.ac.il/~gideon/mturk771.html

[60] B. Gao, J. Bian, and T.-Y. Liu, "Wordrep: A benchmark for research on learning word representations," 2014.

[61] A. Almuhareb, "Attributes in lexical acquisition /," p. University of Essex, 01 2006.

[62] M. Baroni and A. Lenci, "How we BLESSed distributional semantic evaluation," in *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. Edinburgh, UK: Association for Computational Linguistics, Jul. 2011, pp. 1–10. [Online]. Available: https://www.aclweb.org/anthology/W11-2501

[63] W. Battig, W.F. Montague, "Category norms for verbal items in 56 categories: A replication and extension of the connecticut norms," 1968.

[64] M. Baroni, S. Evert, and A. Lenci, "Esslli workshop on distributional lexical semantics bridging the gap between semantic theory and computational simulations," 2008.

[65] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," 2017.

[66] H. Lim, D. G. Andersen, and M. Kaminsky, "3lc: Lightweight and effective traffic compression for distributed machine learning," 2018.

TRITA-EECS-EX-2020:484