Quantum Algorithms for Compositional Distributional Semantics in Natural Language Processing

Master Thesis, Utrecht University, January 2023

Vittorio Pagni

Universiteit Utrecht

Supervisors:

PROF.DR.IR. H.T.C. STOOF , PROF.DR. M. MOORTGAT

Contents

1 Acknowledgements **5**

2 Introduction **6**

3 Different Approaches to NLP **7**

3.1 Three of the most famous paradigms 7

3.1.1 Advantages and Disadvantages 7

3.1.2 Overlaps 8

3.1.3 Our approach 8

3.2 What is a neural network? 8

3.3 The Artificial Neuron 9

3.4 The Supervised Training Procedure and the Backpropagation Algorithm 10

4 Distributional Semantics **10**

4.1 Neural Networks applied to Distributional Semantics 10

4.1.1 One-hot Vectors and Embedding Layers 11

4.1.2 Word2Vec 11

4.1.3 CBOW 13

4.1.4 Skip-gram 13

4.1.5 Recurrent Neural Networks 14

4.1.6 Transformers 15

5 Types of sentences **18**

5.1 Types of Declarative sentences 18

6 Ambiguities in Natural Language **18**

6.1 Types of Ambiguities 19

6.2 Ambiguities of a sentence - Pronoun Substitution 20

6.3 Application of Sentence Similarity to the Pronoun Substitution Problem:
A qualitative analysis 20

6.3.1 Interpretation of the results 23

6.3.2 GPT3 24

7 How to simulate the action of adjectives on nouns using matrices 24

7.1 Compositionality, contextuality and categorial grammars 24

7.1.1 Compositional Semantics 24

7.1.2 The Principle of Contextuality 25

7.1.3 Categorial Grammars and Lambek calculus 25

7.2 Adjectives as transformations on the noun space 26

7.2.1 Previous contributes 26

7.2.2 How to build a representation for the adjective map 28

7.2.3 The image vectors 28

7.3 DBScan Clustering Algorithm and the problem of dishomogenous density 29

7.3.1 A possible solution 33

7.3.2 Procedure Results 35

7.4 Using clusters to train adjective representations 36

7.5 The problem of biases in the data set 37

7.5.1 Norm of Vectors 41

7.6 Results of the procedure 43

7.7 Further improvements 43

7.8 Adapting this approach to rotations 43

7.8.1 The initial approach: Lie Groups 43

7.8.2 A help from the literature: The Orthogonal Procrustes Problem 46

7.9 Incorporating Transformers into the procedure 46

7.10 Region of Action of Adjectives 47

7.11 Visualizing the action of the adjectives on the cluster centers 48

7.11.1 Graph Representation 48

7.11.2 Multi-dimensional Scaling (MDS) 48

7.11.3 Distance Distribution 50

7.11.4 K-Means 50

7.11.5 PCA Analysis and Vagueness of an Adjective 51

7.12 Local Linear approximation as a possible approach 54

7.12.1 Internal structure of the noun clusters 54

7.12.2 A candidate for the transformation in 3D 54

7.12.3 A model for N dimensions: The Procrustes Problem and The Kabsch's Algorithm 55

7.12.4 Local Rotations 57

7.12.5 Studying the case of a linear angle transformation 59

7.12.6 2D Visualization 62

7.12.7 Mix of Von Mises-Fisher Distributions 64

8 Derivation Trees **67**

8.1 Categorial Grammars 67

8.2 Ambiguities 67

8.3 A classical attempt at a solution to the derivation tree problem 68

8.3.1 The algorithm 71

8.3.2 Beyond the brute-force approach 75

8.3.3 Compositional sistributional semantics using tensor contraction 76

8.4 A quantum version of this algorithm 77

8.4.1 Encode classical information into quantum states 77

8.4.2 A quantum circuit to contract indices . . . 78

8.4.3 An alternative to the permutation operator . . . 81

8.4.4 Generalization to N dimensions . . . 82

8.4.5 The optimal number of dimensions . . . 85

8.4.6 The Loss of Amplitude . . . 86

8.4.7 How to contrast the loss of amplitude . . . 86

8.4.8 Cost Analysis of the Amplitude Amplification . . . 88

8.4.9 Simultaneous readings . . . 89

8.4.10 Comparing states . . . 92

8.4.11 Contractions between vectors and rotations . . . 95

8.4.12 How to incorporate grammar rules . . . 97

8.4.13 Three index tensors : AND . . . 98

8.4.14 R-O-L contraction . . . 99

8.4.15 R-O and O-L contractions . . . 100

9 Conclusions **102**

Bibliography **103**

1 Acknowledgements

I would like to express my gratitude to my first supervisor, Prof. Dr. Ir. H. T. C. Stoof, for the patience he showed during our weekly meetings and all the valuable comments and suggestions he offered in support of my work. I also want to thank Prof. Dr. M. Moortgat, my second supervisor, for introducing me to the vast and fascinating area of linguistics and more specifically to the field of formal language theory, where I just took a few steps. Thanks should also go to Dr. Adriana D. Correia, whose PhD thesis on quantum natural language processing deeply inspired and motivated my work, as it often served as a point of reference. I am also grateful to the postdoctoral researcher at Utrecht University, Gijs Wijnholds, who accepted to meet me and gave me useful suggestions on how to analyze and improve my results. I'd also like to congratulate my classmate and friend Leonardo Hügens, who recently finished his thesis on the same topic. I came to the realization that I have mostly been following in his footsteps, and I must say that I respect him both as a scientist and as a person. Unfortunate circumstances prevented me from discussing my job in a large team very frequently. However, I am appreciative of my fellow students, particularly my friend Matteo Massaro, with whom I had significant professional and personal moments while we were both working on our theses. I truly believe in the power of cooperation, and I have discovered the importance of sharing and discussing ideas, as well as carefully listening to others' comments. I would be remiss in not mentioning the deep influence this university and this city have had on me, creating a stimulating and warm cultural environment that never stops surprising me. While seemingly different, the enthusiasm and commitment I saw in the university staff reminded me of my country's large community of intellectuals and passionate researchers, underscoring how the love for acquiring and sharing knowledge still makes us even closer than we already are. Finally, this endeavor would not have been possible without the support of my family, to whom I owe almost everything. They guided me, advised me, and trusted me, and for that I will always be grateful.

Abstract

In this thesis, we try to combine machine learning and quantum computing in order to simultaneously perform different readings of a sentence. In the first part, we introduce a procedure to obtain a distributional tensor representation for adjectives seen as endomorphisms on the noun space. We accomplish this by discretizing the surface of the n-sphere, where the noun embeddings lie, and finding local approximations. We show how the analysis of their activity distributions can give us some information about their action in the noun space. We also provide different options to approximate their local behavior and visually represent their activity. This approach could, in principle, be applied to any kind of transformation in the sense of the types of categorial grammars. The second section, which is devoted to the quantum approach to the problem, explains how these classical representations can be effectively encoded into quantum states. Additionally, it involves the introduction of some quantum circuits, which, when used in conjunction with a traditional machine learning strategy, enable us to simultaneously perform several readings of ambiguous sentences using index contractions and to assign them a likelihood score.

2 Introduction

“Words are, in my not-so-humble opinion, our most inexhaustible source of magic. Capable of both inflicting injury, and remedying it.”

Albus Dumbledore (Harry Potter and the Deathly Hallows by J.K. Rowling).

Natural language processing (NLP) is the ability of a computer program to understand human language as it is spoken and written. It is a component of artificial intelligence (AI) and also a meeting point between this field and the ones of computer science and linguistics. The importance of teaching a computer how to extract and process information from a corpus was first underlined by the father of computer science, Alan Turing, who published an article titled ”Computing Machinery and Intelligence” in 1950 where he included human text interpretation as the crucial part of his famous test for artificial intelligence. Although different animal species are able to communicate relatively complex information through different means [AG69], abstract language is predominantly a human feature and it is the foundation of our articulated thought and rich social interactions. NLP has currently several applications in everyday life that go from speech recognition to typing suggestions for text messages or sentiment analysis of web pages. This field of research has been dominated by deep neural networks in the last ten years, mainly due to the constantly increasing power of computers (Moore’s law) and the significant growth of available data on the world wide web to train these programs on. The state-of-the-art neural models, for example Transformer, can achieve surprisingly good results on different NLP tasks as automatic text generation, question answering or sentence comparison. Nevertheless, these architectures have less solid theoretical basis than other approaches and are intrinsically difficult to interpret by humans. Additionally, the severity of their mistakes is not always proportional to the difficulty of the task, with machines sometimes failing to solve problems that seem easier from a human perspective, while successfully tackling more complex problems in other cases. One possible explanation is that the human mind is capable of performing highly complex tasks, such as creating and interpreting language, recognizing faces and objects, and coordinating hundreds of muscles, all while constantly adapting to external stimuli. All of these features appear trivial, so that we always take them for granted whenever we carry out our daily activities, but they revealed themselves in all their complexity once scientists tried to teach computers how to perform even their simplified versions. Despite this, the field of NLP research is active and profitable, and neural networks have seen significant improvement in a short period of time. For example, AI can now write piano fugues in D, create new artworks in the style of Salvador Dalí and perform rapid translations and intelligent information retrieval on large amounts of text.

In this thesis we focus our attention on one of the most fascinating and useful tools of our species: the natural language, specifically English. Although most of the concepts we will introduce can be generalized to the majority of currently spoken languages, this operation is always interesting and far from easy, as it often occurs whenever we tackle a difficult and rich problem. Our main goal is to provide a concise and general (albeit far from complete) overview of some methods that have been applied to NLP in the past 20 years, which are the heirs of centuries of mathematical and philosophical study of human language. We will also suggest some potential improvements that could be achieved with a quantum computing approach, in particular in the context of compositional semantics. In the first few chapters, we introduce some of the most well-known paradigms in NLP, as well as provide an overview of the basic concepts of the logical analysis of English sentences.

3 Different Approaches to NLP

3.1 Three of the most famous paradigms

There have been several frameworks in NLP, which have split and unified multiple times across history. Among the main paradigms we find the ruled based, the statistical and the neural network approaches, which have significant overlaps and have alternatively been under the spotlights during the last 80 years. Historically, the problem of analyzing natural language and finding rules within its structure has been tackled by different disciplines. These efforts have gradually focused on related sub problems, assuming several names such as computational linguistics in linguistics, natural language processing in computer science, speech recognition in electrical engineering, computational psycholinguistics in psychology and so on. The symbolic (or ruled based) and the statistical (or stochastic) paradigms were the first to appear as heirs of a prolific period during and especially right after World War II, when formal language theories and information theoretic models were developed. The first set includes Chomsky's context-free grammar (1956) for natural language, whereas the second is strongly based on Shannon's works on different topics such as information entropy, communication channels and data transmission. Even the foundations of the neural network approach were laid during the same period, with McCulloch-Pitts' neuron appearing in 1943 as the first simplified computational unit directly inspired by the brain. However, it was only with the diffusion of sufficiently powerful computers and large amounts of available data that this bio-inspired mathematical model led to the first artificial neural networks, which completely changed the state of the art. [DJ23].

3.1.1 Advantages and Disadvantages

One of the main advantages of the symbolic models is their interpretability, since they are by construction relatively readable, clean and logical. Furthermore, because they generally work at higher level of abstraction, this class of models can borrow concepts and techniques from linguistics and logic, exploiting decades of experience and powerful mathematical tools such as Set theory and more recently Category Theory. These systems can also be programmed to tackle specific and infrequent tasks in NLP, for example rare grammar constructions or out of vocabulary words, with an accuracy level that can be superior to other approaches. On the other hand, statistical models and especially deep neural networks often require a huge amount of training examples, as they gradually learn how to solve a specific task, and their learning rate decreases significantly as soon as they reach an high level of accuracy . This weakness is nevertheless also the source of their superior versatility and adaptability, since they can be fine tuned for a specific task. They also possess the ability to tackle problems which are difficult or even practically impossible to model precisely with a rigid set of rules, for example object and face recognition. Moreover, differently from ruled based approaches which tend to become extremely complicated as they become more and more general, increasing the number of neurons and training examples, modifying the network structure and setting a few hyper-parameters is often enough to enhance the generalization ability of a neural network.

3.1.2 Overlaps

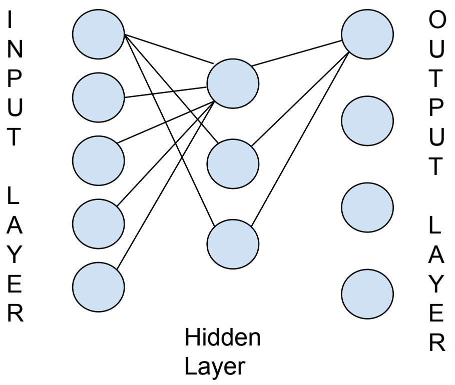
The overlaps between these methods are frequent. For example in the automatic translation of text, which can be achieved using statistical machine translation (SMT) algorithms, the machine analyzes existing translations between two given languages and then defines rules that are the most suited for translating a particular sentence. After SMTs, several more advanced statistical methods have been introduced through the last twenty years, for instance the distributional models which analyze the relative frequency of n-grams in the corpus. In the relatively recent period the analogous neural approach, called Neural machine translation (NMT), has become the most widely used as its performance results superior for some applications, especially when dealing with under-resourced languages and where the context plays a main role. [Dow+18].

3.1.3 Our approach

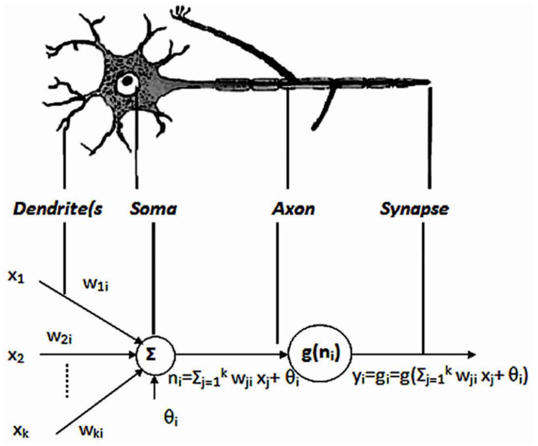
As it often occurs, in this thesis we used a mix of these paradigms. Starting from a set of word embeddings, which had been learnt using a neural network, we followed the approach of Compositional Semantics and looked at the different syntactic elements of a sentence, such as adjectives or adverbs, as transformations that act on the elementary units of meaning. These units are words or small groups of words belonging to the same syntactic type, for example nouns, that we consider as the basic carriers of the information in the text. In this case, the rules of grammar tell us the type of transformation that each word of group of words performs on the other and the order of their application. Our analysis focuses in particular on the action of the adjectives on nouns, studying the way the presence of the former modifies the vector representation of the latter.
All of the concepts we have introduced in these few lines will be briefly explained or refreshed in the next sections.

3.2 What is a neural network?

A neural network is a type of machine learning model that is inspired by the structure and function of the human brain. It is a mathematical structure which is composed of layers of interconnected nodes called artificial neurons. Each node receives inputs from an arbitrary amount of other nodes, weights this signals according to its own internal parameters and produces an output. The output of one layer of nodes becomes the input for the next layer, and the final output is produced by the output layer. Neural networks are trained using a set of labeled examples, which consists of input data and the corresponding correct output. The training process involves adjusting the weights and biases of the connections between the nodes to minimize the error between the predicted output and the correct output.



(a) Scheme of a simple neural network. (Not all of the connections have been drawn.)



(b) Comparison between a biological and an artificial neuron. (the image is taken from the article [BBP16])

3.3 The Artificial Neuron

An artificial neuron or Perceptron [58], is a mathematical model of the biological neuron, which produces an output signal based on n inputs, a correspondent set of weights W, bias vector **b** and an activation function. The latter is often chosen to be a logistic sigmoid or a rectified linear unit function.

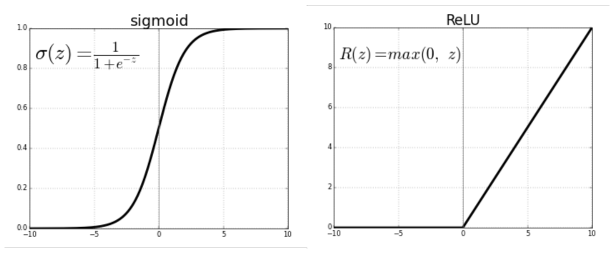


Figure 2: Plots of the two most used activation function, taken from Activation Functions in Neural Networks by Sagar Sharma `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6`.

These neurons, similarly to the their biological counterparts, process and transmit information and each of them performs a relatively simple task, which is modeled by the following expression:

$$y_k = g(n_k) = g(\sum_i W_{k,i} x_i + b_k) \tag{1}$$

Neural networks can be used for a wide range of tasks, including image classification, language translation, and speech recognition. They are particularly useful for tasks that require the ability to learn and adapt from data, as they are able to learn complex patterns and relationships in the data.

3.4 The Supervised Training Procedure and the Backpropagation Algorithm

On of the most important features of neural networks is that they need a training phase, when the model is exposed to a large amount of input data and its answers are compared with the correspondent correct outputs, provided by the user. The quality of the predictions and hence the performance of the model is inversely proportional to the value of a cost function, calculated between the predictions of the output layer and the correct output. An example of cost function is the Means Squared Error function:

$$C_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{2}$$

Where: n is the total number of examples in the data set y_i is the true output for example i $\hat{y_i}$ is the predicted output for example i
Using an optimization algorithm such as the Stochastic Gradient Descent [Rud16], we can repeatedly adjust the weights and the biases of the network, which are the only free parameters and represent the current knowledge of the model, in order to minimize the cost function. The network is composed of several layers which act sequentially and the cost function is only directly related to the value of the last (output) layer. For this reason, in order to calculate the right adjustments for the weights of the more internal neurons, we need to apply a famous algorithm called Backpropagation, which is essentially based on the repeated application of the chain rule to the derivative of the cost function with respect to the weights and the biases. The reader can find more details about the Stochastic Gradient Descent and the Backpropagation algorithms here [GBC16].

4 Distributional Semantics

Distributional Semantics is a research area in NLP which is fundamentally based on the following hypothesis:
“The degree of semantic similarity between two linguistic expressions A and B is a function of the similarity of the linguistic contexts in which A and B can appear.” [Har54]
This assumption doesn’t always reveal correct, for instance because words with opposite meaning like ”hot” and ”cold” tend to appear in the same context. Despite these limitations, describing a word by the company it keeps can be a powerful principle on which it is possible to build a vector representation of words and sometimes even sentences.

4.1 Neural Networks applied to Distributional Semantics

If we want to quantitatively describe the meaning of words and their relations, a powerful approach is to represent them as vectors, called word-embeddings, which spatially capture their similarities. There are different ways to do so and, as we will see, some are more efficient than others. Neural networks can efficiently be trained to build such representations using dense embeddings of real vectors, which can be used to perform several NLP tasks such as word and sentence comparison, question answering and sentiment analysis. In the following subsections we introduce a famous model based on neural networks, Word2Vec, as well as its main components. Although quite powerful for building accurate word embeddings, this type of approach doesn't give particular importance to the order of the words in a sentence and it's therefore not optimal to represent long portions of text. Since a significant part of the information is encoded in the order, scientists developed a new kind of neural network architecture which is specialized in processing sequential inputs: the recurrent neural networks or RNNs [RHW86]. Although RNNs in all their variants showed to be a powerful model [Sch15], due to some limitations of the training procedure of large networks, their performance has been surpassed in many field by transformer architectures.

4.1.1 One-hot Vectors and Embedding Layers

An embedding layer is a neural network layer that converts discrete, one-hot encoded vectors into continuous, dense vectors. One-hot vectors are vectors with a single non-zero element, typically used to represent categorical variables such as words in a vocabulary, they therefore have the size of the vocabulary or of a subset. In contrast, dense vectors are vectors with multiple non-zero elements, usually real, which can capture more complex relationships between the elements of the vector. Given a vocabulary containing N words and a N-dimensional one-hot vector representing the word "cat", which has a one in correspondence of the position of the word cat in the vocabulary and zero elsewhere, we can use embedding layer to map this huge sparse vector into a dense vector with real entries. The length of the dense vector is usually much smaller than the size of the dictionary, typically a few hundreds of entries, but it still captures the meaning of the word "cat" in a continuous space.

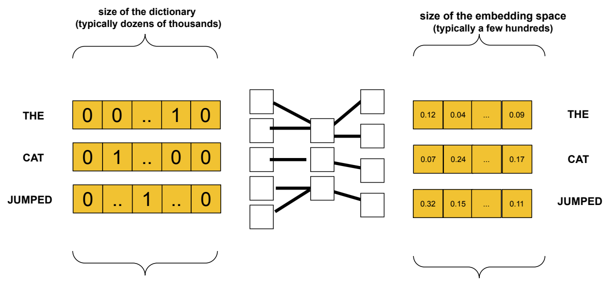


Figure 3: Simplified scheme of an embedding layer

[Mik+13a]

4.1.2 Word2Vec

Word2Vec is a model for learning continuous vector representations of words from large amounts of unstructured text data. It was developed by researchers at Google in 2013 [Mik+13b] and has since become widely used in natural language processing tasks such as language modeling, machine translation, and text classification . Word2Vec has been shown to be effective at learning high-quality word embeddings that capture certain semantic and syntactic relationships between words. For example, in a trained Word2Vec model, similar words tend to have similar vector

representations and dissimilar words tend to have dissimilar vector representations. This allows the model to capture the meaning of words in a continuous, low-dimensional space and to perform tasks such as analogy completion (e.g. ”man is to woman as king is to X”).
One of the key advantages of Word2Vec is that it can learn word embeddings from large amounts of unstructured text data without the need for manually-annotated labels or extensive preprocessing. This makes it a valuable tool for a wide range of natural language processing tasks.
There are two main ways that Word2Vec can be used to learn word embeddings: continuous bag-of-words (CBOW) and continuous skip-gram.

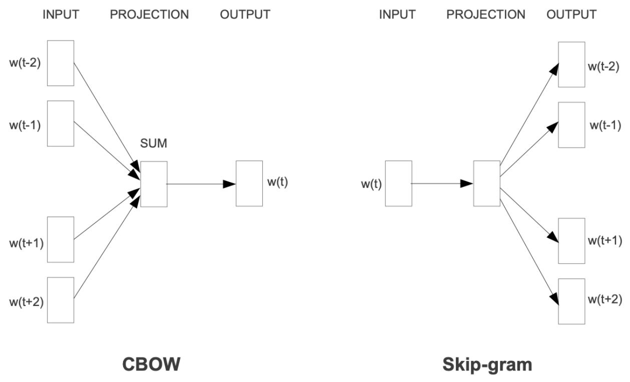


Figure 4: Diagram of the architecture of CBOW and Skip-gram taken from the article
”Efficient Estimation of Word Representations in Vector Space”[Mik+13b]

In CBOW, the model takes a context window of words surrounding the target word and predicts the target word given the context. The model is trained to minimize the cross-entropy loss between the predicted probability distribution over words and the true distribution, which is obtained by treating the target word as the true class and the context words as the negative classes.
Conversely, continuous skip-gram takes a target word and predicts the context words surrounding it. This time the model tries to maximize the average log probability of the context words given the target word.
In summary, both CBOW and continuous skip-gram are based on the idea of predicting a word given its context, but they differ in the way that the context is defined and in the direction of the prediction. C predicts the target word given the context, while continuous skip-gram predicts the context given the target word.

4.1.3 CBOW

The continuous bag-of-words (CBOW) model is a neural network-based method for predicting a target word given the context words in a sentence. It is commonly used for natural language processing (NLP) tasks such as language modeling and word embedding. In the CBOW model, the input to the network is a set of context words surrounding the target word, and the output is a probability distribution over the words in the dictionary to be the target word at the center of the context window. The context words are typically represented as one-hot vectors, which are converted to a dense representation using an embedding layer. The dense representation of the context words is then averaged and passed through a fully-connected (FC) layer, followed by a softmax layer to predict the target word.

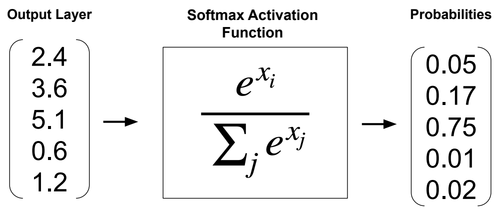


Figure 5: Softmax Activation Function

The CBOW model is trained to minimize the cross-entropy loss between the predicted distribution on the target words and the actual distribution, which is a delta.

$$Loss_{CE} = -\sum_{i}(p_i)\log(q_i)) \tag{3}$$

where q_i is the predicted probability distribution over classes and p_i is the true probability distribution. The sum is taken over all classes. During training, the model is presented with a sequence of context-target word pairs, and the weights of the model are updated using backpropagation and an optimization algorithm such as stochastic gradient descent (SGD).

4.1.4 Skip-gram

The Skip-Gram model performs a task which is in a certain sense the opposite of CBOW's: given a certain word in the dictionary, represented by a one-hot vector of dimension $\mathcal{V}$, the network outputs another vector of the same length whose entries represent a probability distribution over all the words in the dictionary to appear in the context of the input word. In the paper [Mik+13b], Google's researchers used a hidden layer of 300 neurons representing 300 features that the network uses to model the relation between a word and its context. The weights of the network are again learnt by supervised training using multiple examples of pairs (word,context) and the gradient descent algorithm. In order to improve the training procedure of the model, the authors also used negative sampling. This is a technique that speeds up training processes that involve a huge amount of non-target examples, in our case all the words in the dictionary that don't belong to the specific word-context pair example. The basic idea is that instead of using all the non-target examples (i.e., negative examples) to update the model's parameters during training, a small subset of them is randomly sampled (according to a specific distribution) and used instead. This greatly reduces the computational cost of training the model while still allowing it to learn effectively from the data.

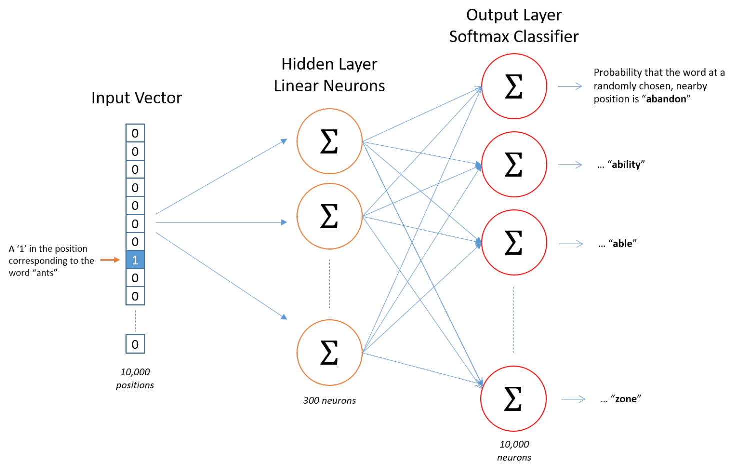


Figure 6: Scheme of the architecture of the Skip-Gram model. The figure is taken from [Chr17], which in turn cites [Mik+13b].

4.1.5 Recurrent Neural Networks

These models posses a hidden layer, which allows them to store information about the sequence of the words. The RNN processes the data one step at a time using the hidden state of the previous step as input for the current step. This recursion, which is again inspired by the connections in the human brain, allows the network to capture the dependencies between the elements in the sequence and to use this information to make predictions. In the picture below we illustrate the scheme of a simple model of RNN, named after Jeffrey Elman, where there is one hidden layer which takes is current state as input as well as the activation value of the hidden layer from the previous time step 7. Despite their general flexibility, RNNs can be difficult to train and can suffer from the vanishing gradient problem, which makes it hard for the gradients to flow through the network and update the weights [JM00].

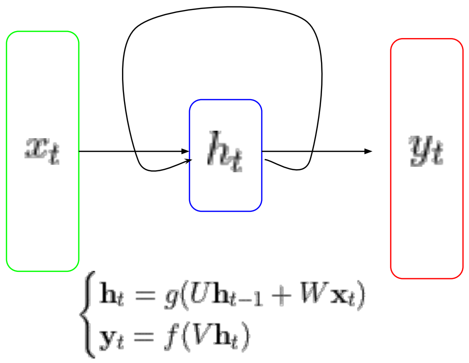

x_t
h_t
y_t
$\begin{cases} \mathbf{h}_t = g(U\mathbf{h}_{t-1} + W\mathbf{x}_t) \\ \mathbf{y}_t = f(V\mathbf{h}_t) \end{cases}$

Figure 7: Simple recurrent neural network after Elman [Elm90](1990). The hidden layer includes a recurrent connection as part of its input. That is, the activation value of the hidden layer depends on the current input as well as the activation value of the hidden layer from the previous time step through, respectively, the two matrices U and W.

4.1.6 Transformers

Transformer models, on the other hand, are a type of neural network that is designed to process sequential data using an attention mechanism. They do not use the hidden state of the previous step as input for the current step, but instead use attention to weight the input elements and to determine which input elements are most important for the current step. This allows transformer models to parallelize the computation, which makes them faster and easier to train than RNNs. After the publication of the famous paper ”Attention is all you need” [Vas+17], which introduced the self attention mechanism, these models have gradually achieved state-of-the-art results on a wide range of natural language processing tasks and have become the default choice for many NLP tasks, such as language translation, image captioning, and text summarization. Their architecture is composed of an encoder and a decoder, with the encoder being responsible for analyzing the input and the decoder for producing the output. The encoder and decoder are both made up of layers of multi-head self-attention and feed-forward neural networks. The multi-head attention allows the model to attend to different parts of the input at different positions, while the feed-forward neural network allows the model to make use of the information that has been gathered. As explained in the article and in the online course [Jay], the self-attention mechanism can be summarized by the following steps.

- 1) For each word embedding $\mathbf{n_i}$, the model creates a Query vector $\mathbf{q_i}$, a Key vector $\mathbf{k_i}$, and a Value vector $\mathbf{v_i}$. These vectors are created by multiplying the embedding by three matrices that are trained during the training process, namely W_Q, W_K and W_V. The sizes of these vectors can also be smaller than the one of the original noun embedding in order to limit the computational cost of the procedure. If we therefore consider the matrix N with the original word embeddings related to a sentence as rows, the model produces the new matrices Q, K and V as

$$\begin{cases} Q = N \cdot W_Q \\ K = N \cdot W_K \\ V = N \cdot W_V \end{cases} \tag{4}$$

- 2) The output of the self-attention layer is given by the matrix

$$Z = softmax(\frac{Q \cdot K^T}{\sqrt{d_k}}) \cdot V \tag{5}$$

  where d_k is the size of the key vectors.

The paper then refines this procedure by introducing the concept of multi-headed attention. This process is repeated several times (usually in parallel), using a different set matrices W_{Qi}, W_{Ki} and W_{Vi} each time. At the end, all the resulting matrices Z_i are concatenated in a larger matrix $Z = (Z_1, ...Z_n)$ and multiplied by a large weight matrix W. The result of this multiplication is a final matrix Z_F of the same dimension of the Z_i. This matrix represents the attention matrix at a given encoding or decoding step.

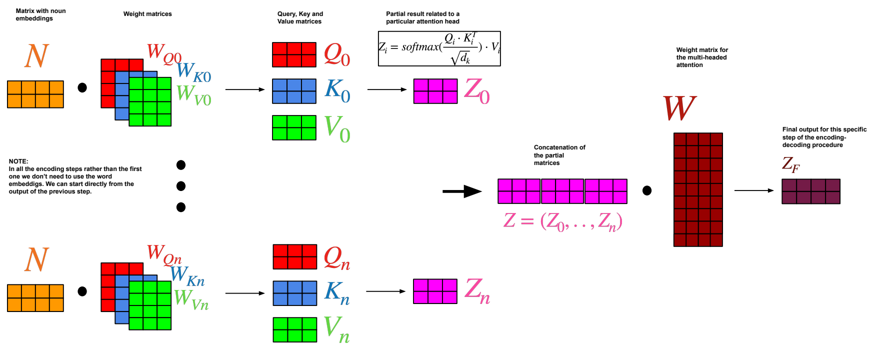
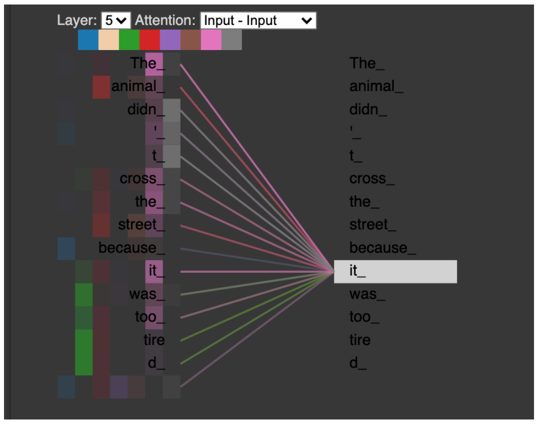
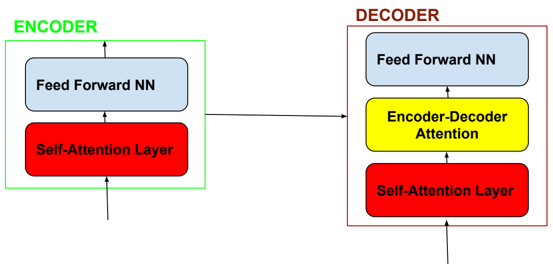

Matrix with noun embeddings
Weight matrices
Query, Key and Value matrices
Partial result related to a particular attention head
$Z_i = softmax(\frac{Q_i \cdot K_i^T}{\sqrt{d_k}}) \cdot V_i$
N
W_{Q0}
W_{K0}
W_{V0}
Q_0
K_0
V_0
Z_0
NOTE:
In all the encoding steps rather than the first one we don't need to use the word embeddigs. We can start directly from the output of the previous step.
Concatenation of the partial matrices
$Z = (Z_0, . . , Z_n)$
Weight matrix for the multi-headed attention
W
Final output for this specific step of the encoding-decoding procedure
Z_F
N
W_{Qn}
W_{Kn}
W_{Vn}
Q_n
K_n
V_n
Z_n

Figure 8: Illustration of the multi-headed attention mechanism.



(b) Example of the self-attention mechanism applied to the sentence "The animal didn't cross the street because it was too tired." We can see that in the last encoder unit in the stack, in this case the 5th, while encoding the word "it" part of the attention mechanism was focusing on "The Animal", and baked a part of its representation into the encoding of "it". The image is taken from the colab notebook Tensor2Tensor, which is used and maintained by researchers and engineers within the Google Brain team and a community of users.`https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb#scrollTo=odi2vIMHC3Rm`



(a) Simplified scheme of the encoder and decoder units of a transformer model. The model usually stacks multiple encoder and decoder units.

In summary, RNNs and transformer models are both types of neural networks that can be used for natural language processing tasks, but they have different architectures and the way they process sequential data. RNNs use the hidden state of the previous step as input for the current step, while transformer models use attention to weight the input elements and determine which input elements are most important for the current step.

5 Types of sentences

There are four types of sentences in English:

- **Declarative Sentences** ex. ”I go to school.”
- **Imperative Sentences** ex.”Go to school.”
  They deliver an instruction or an order.
  They usually end the period and may be followed by an exclamation point.
- **Interrogative Sentences** ex. ”Who goes to school?”
  They ask a direct question and therefore have a question mark at the end.
- **Exclamatory Sentences** ex. I go to school!
  They are used to underline a strong feeling or emotion and are always followed by an exclamation mark.

In this thesis we will focus on the first type, which is also the most common one.

5.1 Types of Declarative sentences

The set of declarative sentences divides into :

- **Simple Declarative Sentences** ex. ”I usually wake up early”.
  They do not contain dependent sentences but only one independent sentence.
- **Compound Declarative Sentences** ex. ”I usually wake up early but I always have breakfast.”
  They do not contain dependent sentences but two or more independent sentences.
- **Complex Declarative Sentence or period** ex. ”I usually wake up early because I have to.”
  They contain only one independent sentence and one ore more dependent ones.
- **Compound-Complex Sentences** ex. ”I usually wake up early because I have to, but I always have breakfast.”
  They have one or more independent sentences and at least one of them has one or more dependent sentences.”

6 Ambiguities in Natural Language

Natural languages are extremely powerful, as they can more or less efficiently express any concept we think, write or read about. However, the decoding of a written or spoken message is a delicate and complex process, which often gives rise to ambiguity. Our brains are extremely good at dealing with them, automatically selecting the most likely interpretation for a given phrase or sentence, so that we are not even aware of their presence, unless it is deliberately accentuated as in poetry. There are several types of ambiguities that happen at different level of the decoding procedure. Here we show some examples:

[Kit]

”Well, I’ve certainly never tasted chicken cooked that way before!”
Was the chicken good or bad?

”Call me a taxi, please.”
Is the speaker asking someone to hail them a taxi or to be called a taxi?

”Stop trying to push the envelope.”
Is someone trying to push the boundaries in a current situation or literally push an envelope across a desk?

”I saw someone on the hill with a telescope.”
Did you use a telescope to see someone on the hill, did you see someone on the hill holding a telescope or where you on the hill when you saw someone with your telescope?

6.1 Types of Ambiguities

- **Syntactic Ambiguities of Words**
  The same word can belong to multiple syntactic categories: ex. run (VERB) VS run (NOUN)
  ex. fast (NOT EAT) VS fast (QUICK) VS fast (PERIOD WITHOUT EATING)

- **Semantic Ambiguities of Words**
  a) Different meanings inside the same syntactic category:

  NOUNS ex. bank (FINANCIAL INSTITUTION) VS bank (OF A RIVER)
  ADJECTIVES ex. fast (QUICK), fast (FIXED)
  VERBS ex. break (DAMAGE) break (END)
  ADVERBS ex. trying ”hard” (WITH EFFORT) VS raining ”hard”(A LOT)

  b) The meaning is intrinsically vague.

  Adjectives:—ex. tall, fast, big (gradable adjectives).

- **Ambiguities of a sentence**
  a) Pronoun Substitution Problem
  ex. ”the mother loved her daughter because she obeyed her”

  b) Different Word Contractions—
  ex. ”((rigorous mathematicians) and physicists).” vs ”(rigorous (mathematicians and physicists)).”

  c) Words as a string of text or as meaning carriers ex. ”Call me a taxi” vs ”Call me ’a taxi’ ”

6.2 Ambiguities of a sentence - Pronoun Substitution

This type of ambiguity comes from the fact that, whenever a pronoun acts as a placeholder for a noun, multiple possibilities might be grammatically correct. However, it is usually quite natural to understand which one is most reasonable substitution from the context. For example:
”The mother loved her daughter because she obeyed her ”
vs
”The mother loved her daughter because she bore her ”

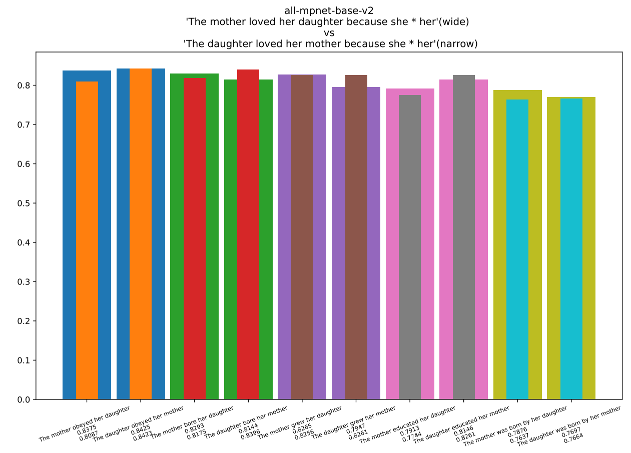
Which have as reasonable substitutions respectively:
she $\rightarrow$ daughter ; her$\rightarrow$mother
she $\rightarrow$ mother ; her$\rightarrow$daughter

6.3 Application of Sentence Similarity to the Pronoun Substitution Problem: A qualitative analysis

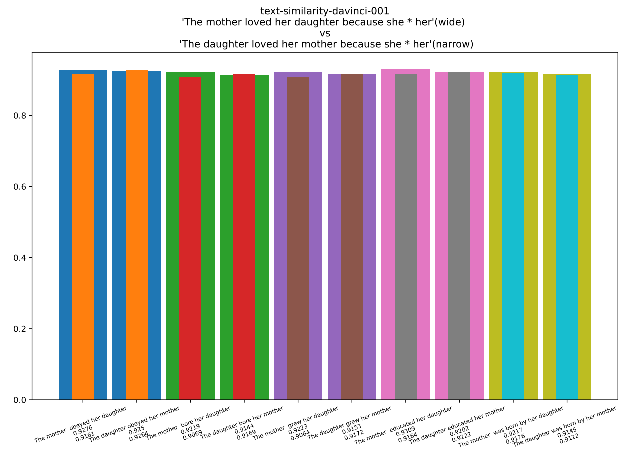
In this example we took two models based on Sentence Transformers, BERT ”all-mpnet-base-v2” and GPT ”text-similarity-davinci-001”, and we used their sentence embeddings to tackle the pronoun substitution problem and passive form one. We repeated the experiment twice: the first time we used words which are semantically (and hence distributionally) quite strongly related to each other and with the verb of the sentence, such as ”mother” ,”bore” and ”daughter” or ”thief” ”arrested” and ”policeman”, whereas the second experiment didn’t involve words with a relevant distributional relation, such as verbs like ”greeted” or common names like ”John” and ”Bob”. The test related to the pronoun substitution problem consisted in comparing a given complex sentence, with an independent and a subordinate clause containing two ambiguous pronouns, to two alternative sentences corresponding to the two possible substitutions for the ambiguous pronouns. The second test involved passive sentence because in such a phrase the order of the words is reversed with respect to the meaning of the action. In this type of sentences grammar plays a major role in decoding the meaning .

- Case when there is a strong distributional relation with the verb:
  *∈[”obeyed”,”bore”,”grew”,”educated”,”was born by”](the average product of the cosine distances of the three vectors noun1,verb,noun2 is 0.32 according to the data set of section 7.10):

  Active sentence :
  In this test the target was: ”The mother loved the daughter because she * her.”
  and the options were : ”the mother * her daughter” , ”the daughter * her mother”. We also tried to swap subject and object to study the impact of the word order.



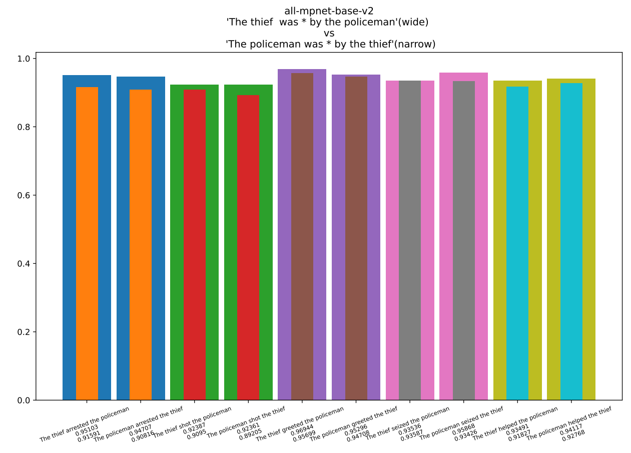
(a) Analyisis of the pronoun substitution problem using model 'all-mpnet-base-v2' based on sBERT



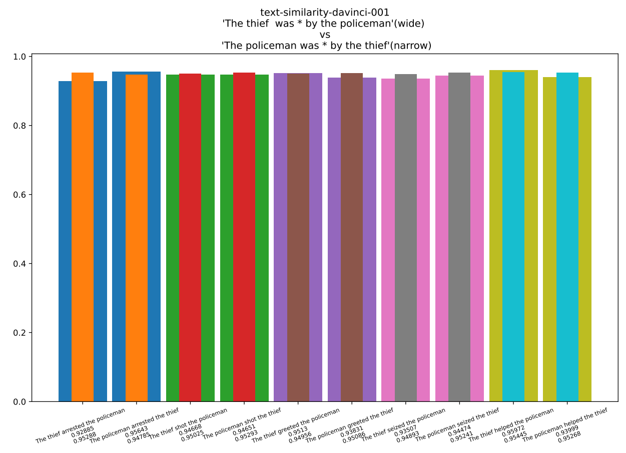
(b) Analysis of the pronoun substitution problem using model 'text-similarity-davinci-001' based on GPT

Figure 10: In the plot the wider histogram represents the first version of the sentence while the narrower one is related to the one with inverse ordering. On the x axis we have the labels of the test sentences and below them the correspondent dot products with the target sentences (the first version above, the second below)
.

Passive Sentence
"The thief was * by the police man." .

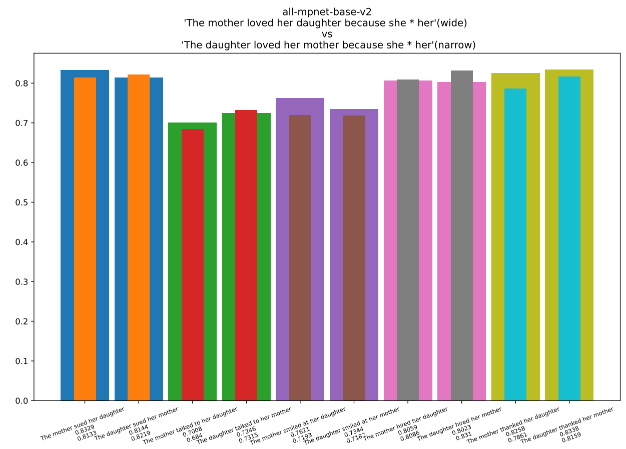


(a) Analysis of the pronoun substitution problem using model 'all-mpnet-base-v2' based on sBERT

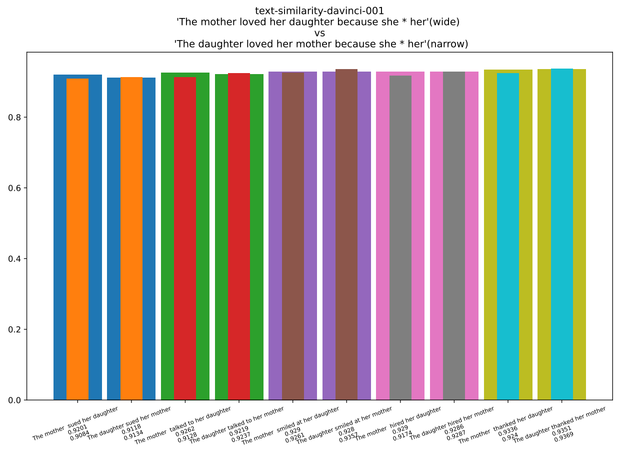


(b) Analysis of a passive sentence using model: 'text-similarity-davinci-001' based on GPT

- Case when there is a weaker distributional relation with the verb:
  *∈[”sued”,”talked to”,”smiled at”,”hired”,”thanked”](the average product of the cosine distances of the three vectors noun1,verb,noun2 = 0.23 according to data set of section 7.10):
  Active sentence:

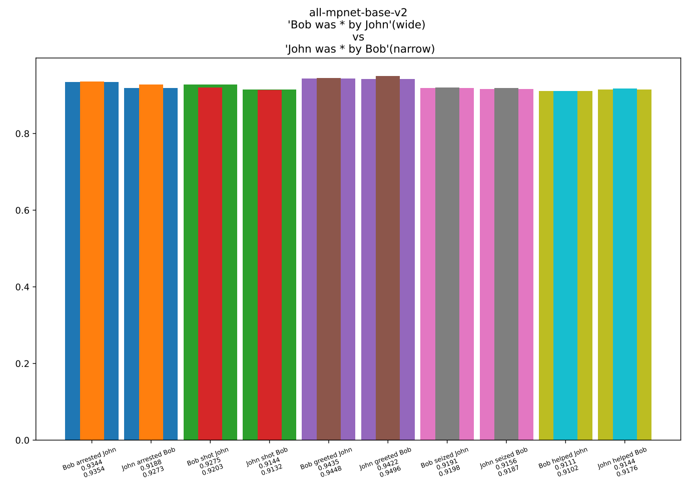


(a) Analysis of the pronoun substitution problem using model ’all-mpnet-base-v2’ based on sBERT (weakly related case)

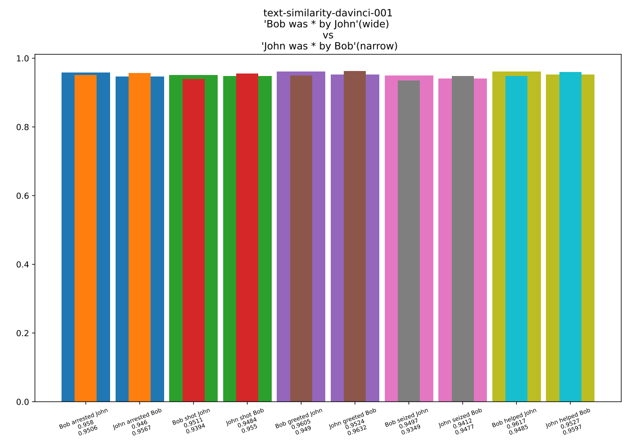


(b) Analysis of the pronoun substitution problem using model ’text-similarity-davinci-001’ based on GPT (weakly related case)

Passive sentence:



(a) Analyisis of the pronoun substitution problem using model: 'all-mpnet-base-v2' based on sBERT (weakly related case)



(b) Analyisis of a passive sentence using model 'text-similarity-davinci-001' based on GPT

6.3.1 Interpretation of the results

From the histograms it is possible to notice how a quite relevant change in the structure of a sentence , like the swap of subject and object, and consequently in its meaning doesn't generally reflect in a significant change in the similarity scores, with an average difference between the two provided interpretation of less than 5 %. This is especially true for the model 'text-similarity-davinci-001', whose answers seem to be weakly influenced by the word order and the correlation between them.
In the case of the active sentences we observe a general tendency of both the models to copy subject-object order of the main sentence when substituting the pronouns in the dependent clause. This feature appears to be more pronounced with 'text-similarity-davinci-001', whereas 'all-mpnet-base-v2' seems to take also the correlation or embedding similarity of words into account, preferring a certain world order in the pronoun substitution regardless of the one given in the main sentence. Furthermore, we notice that, especially for the sBERT model, the similarity between the verb in the dependent clause (the one we substituted to *) and the one in the main clause plays a greater role in the overall similarity between the target complex sentence and the two alternatives.
For what concerns the passive case, the GPT based model seems to provide inconclusive results in the strongly correlated case, with slight and apparently random oscillations of the similarity values while being systematically wrong in the weakly correlated one, displaying a stronger influence of the word order more than of syntactic rules. On the other hand sBERT seems to prefer a certain word order than another, again with relatively tiny differences in the similarity values. Both results are only a qualitative glimpse of the behaviour of model with respect to the syntactic structure of the sentence. From this brief and not rigorous analysis we have an hint on the fact that these two specific model do not seem to have significantly strong reactions to changes in the sentence structure.

6.3.2 GPT3

These two versions are not the state-of-the-art, which could be represented instead by the Generative Pre-trained Transformer 3 (GPT-3), an autoregressive language model that uses deep learning to produce human-like text and can be used as a chat bot [Bro+20]. This AI tool is capable of answering questions, comparing sentences, and automatically creating programming code at a level that has never been seen before. It can also process and interpret vast amounts of text, even language with complex syntax. GPT3's design is a typical transformer network with a few technical modifications and a massive 2048-token-long context. It has 175 billion parameters, which require 800 GB to be stored. Obviously this thesis does not aim to provide a better alternative to this highly effective models, conversely its purpose is to investigate some procedures where the grammatical structure combined with a quantum approach may help to speed up the process of interpreting a sentence and representing its meaning, at least in some areas.

7 How to simulate the action of adjectives on nouns using matrices

One of the most popular and brilliant ideas in NLP is the concept of dense word embeddings, a technique which consists in representing each word in the dictionary (or fragments of words) with a properly normalized vector. This allows to perform algebraic operations on these vectors, such as the application of a linear transformation or the computation of the cosine similarity in order to extract semantic information from a given sentence and even paragraphs. As we explained in the previous sections, there are several ways to build these vectors and a quite efficient one in practice is to use dense arrays of real numbers with unit length, which can be learnt by a model from a corpus. The relative positions of these vectors on the surface of the unit hyper sphere are meant to reflect the semantic relations between the words they represent, so that words that usually appear in the same context in the corpus should also be represented by vectors that are close to each other. Since all of these arrays have unit length, their Euclidean distance directly proportional to their cosine distance:

$$d_{cos}(\mathbf{x},\mathbf{y}) = 1 - Sim_{cos}(\mathbf{x},\mathbf{y}) = 1 - \frac{\mathbf{x}\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = 1 - cos(\theta) = \frac{d_E(x,y)^2}{2} \tag{6}$$

and hence from now on we will consider the latter as our metric. The procedure of building word embeddings can be based on different models, each with its own advantages and weaknesses. Among the most used models in the literature we find Skipgram and CBOW, GloVe and the relatively more recent Transformers, which introduced the self attention mechanism and turned out to be decisively more powerful than their predecessors.

7.1 Compositionality, contextuality and categorial grammars

7.1.1 Compositional Semantics

Compositional Semantics is a branch of Semantics which is based on the (controverted [TMV12]) assumption, called principle of compositionality or Frege's principle [Pel94], which states that the meaning of a phrase can be determined by combining the meanings of its subphrases using rules which are driven by the syntactic structure. The meaning of the latter is in turn obtained from the meaning of the single words and the syntactic structure of the sub-phrase. This approach can in principle recursively construct the representation for any portion of text by means of a set of rules, or functions, that act on the basic constituent or meaning carriers. An example of basic meaning carriers could be given by the nouns, on which adjectives, verbs and prepositions act as transformations to build the meaning of the whole sentence. This approach is therefore a word-to-sentence one, which starting from the single constituents builds a representation for the whole. The field of compositional semantic is quite flourishing and diverse, especially because it is closely related to semantics decomposition, which involves algorithms that decompose the meaning of phrases and concepts into less complex concepts, and to the AI representation of meaning using graphs, a technique that allows programs to visually represent their reasoning by means of graphs relating concepts.

7.1.2 The Principle of Contextuality

Differently than compositionality, according to the principle of contextuality we can only assign a meaning to complete sentences, which define the context in which their constituents occur. We can then follow a sentence-to-word approach to assign a contextualized representation to each word or group of words in the sentence.[ADC22]

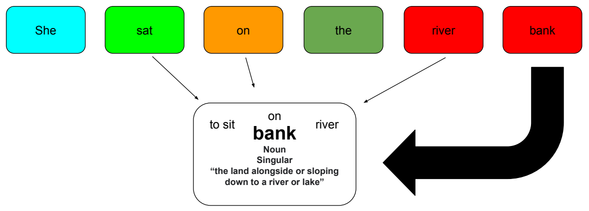


Figure 14: Example of the application of Contextuality to find the correct meaning of the ambiguous word "bank".

7.1.3 Categorial Grammars and Lambek calculus

In formal language theory, a categorial grammar is a grammar formalism in which syntactic category labels are assigned to words and phrases, and the rules of the grammar specify how these labels can be composed to form well-formed phrases and sentences. The key idea behind categorial grammar is that the grammatical structure of a sentence can be derived from the syntactic category of each word and the way in which these categories combine. Some of these categories of words are considered as basic types, for example Noun Phrase, Noun and Sentence (respectively represented as NS,N,S), whereas other types are treated as composite expressions which act as functions from a type to another. For example, adjectives can be considered of type N/N because, whenever they precede a noun, we can look at the latter as their input from the right. This specific contraction of types, namely N/N and N, generates another noun type N as output. An example of categorial grammar is Lambek calculus [Joa58], which was developed by Joachim Lambek in the 1950s and is based on the same basic principles as the lambda calculus-based version of categorial grammar developed by Alonzo Church [Loa98]. Lambek calculus has been used to provide a formal foundation for the study of natural language syntax and semantics, and also used in natural language processing systems. It is known for its mathematical elegance, and its ability to handle a wide range of syntactic phenomena in a uniform way, but it is also criticized for its mathematical complexity.

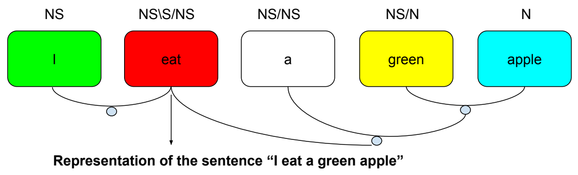


Figure 15: Example of the application of the compositionality using the approach of categorial grammars

7.2 Adjectives as transformations on the noun space

Following the principles of categorial grammars and compositional semantics, together with the ideas introduced in the article [CMS22], we can think of the contraction between and adjective and a noun as a norm-preserving transformation which has the origin as a fixed point and that maps the vector representing a noun into the one associated to the couple adjective plus noun. These two vectors live in the same noun space and hence we are dealing with an endomorphism. If this map had the additional property of globally preserving distances and being surjective we could apply the Mazur-Ulam theorem ([Nic13]) to prove its linearity and therefore the possibility of a matrix representation. However this isn't the case, at least for certain models, whose behaviour does not seem to be linear.

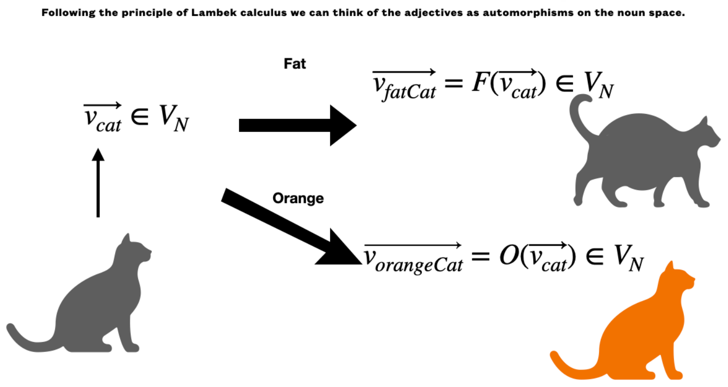

Following the principle of Lambek calculus we can think of the adjectives as automorphisms on the noun space.
Fat
Orange
$\overrightarrow{v_{cat}} \in V_N$
$\overrightarrow{v_{fatCat}} = F(\overrightarrow{v_{cat}}) \in V_N$
$\overrightarrow{v_{orangeCat}} = O(\overrightarrow{v_{cat}}) \in V_N$

Figure 16: Scheme of the action of an adjective

7.2.1 Previous contributes

The idea of considering the adjectives as linear endomorphisms on the noun space has been previously investigated in the article [BZ10], where the authors use linear regression in order to estimate the best matrix representation for a given adjective. This approach differs from ours in the way these matrices are learnt, since our method focuses on exploiting the clustering of the n-sphere in order to locally approximate the adjective map using rotations. One of the reasons we apply this restriction is that local rotations preserve the norm and the distances between the vectors and are therefore suitable to transform quantum states, as we will see in section 8.4.6. Because of this, while the authors of the paper can use the matrix representation in order to globally compare the adjectives, our method provides us with a local similarity score, obtained for example by taking the cosine similarity between the images of the same cluster center under the action of different adjectives.

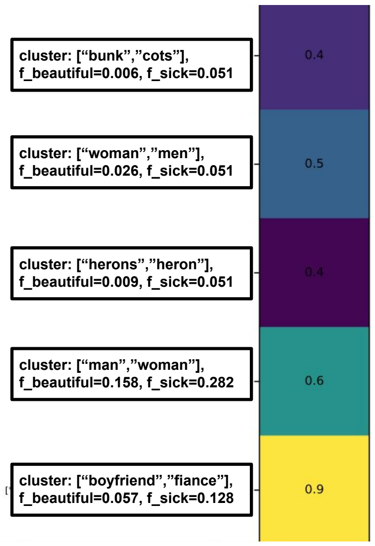


Figure 17: Example of local similarity scores between the adjectives ”beautiful” and ”sick”. The labels indicate the names of the clusters both adjectives are active on. The picture also shows the globally normalized activity frequency of each adjective on the relative cluster. We can notice how the model can sometimes capture the different shades of the meaning of the adjectives as they act on different clusters of words. In this example, although ”beautiful” and ”sick” have in general a clearly different meaning, the fact that the second one can be used as a slang synonym of ”cool” or ”excellent”, especially among young people, can explain the significant increase in the similarity score on the last cluster.

Another more recent contribute is also represented by the article [WSC20] where the authors generalize the Skipgram model in order to learn the representation of certain types Combinatory Categorial Grammar, in particular to represent and compare verbs. Although we only realized the existence of this article later in writing the thesis, we notice a general convergence of ideas. Given the limited amount of computational power and time at our disposal it is difficult to exhaustively compare the two approaches. However, being our approach completely modular, it would be interesting to use representations obtained using this generalized Skipgram model as the classical input for our quantum approach, encoding the tensors learnt from the model into quantum states. Our approach is again slightly different in the way these tensors are learnt as well as in presence of the additional restriction to rotation matrices, since in our procedure we exploit the orthogonality to support certain arguments 8.4.6. This constraint can in principle be loosen, even though this would consequently lead to changes in the amplitude amplification procedure that would then depend on the particular tensors involved in each contraction.

7.2.2 How to build a representation for the adjective map

For what concerns the training procedure, we can in principle treat the bigram composed by adjective and noun as a new independent word, for which the model can build a representation. Some models like sBERT have procedures to build efficient sentence embedding, combining their representation of small portion of text called token and taking into account the context. Although this allows them to capture different shades of the meaning of a word, it makes their single word representations less reliable and more difficult to cluster. For this reason we chose not to work with the word embeddings from BERT but we instead used a data set based on Word2Vec, which we then pre-processed for our purpose 7.5.
In the following sections we will obtain a representation for the transformed word embeddings under the action of the adjectives. Analyzing this vectors we will see that distances, differently than norms, are not globally preserved: the image vectors tend to be significantly closer to each other than the original ones, converging to a number of cluster points which depends on the particular adjective. This shows that the transformation is not linear, and that it therefore cannot be represented, at least globally, as a point-independent matrix. As we will see in the following section, it is sometimes possible to locally approximate its behaviour with a linear operator.

7.2.3 The image vectors

In order to study the action of the adjectives on nouns, which can in principle be a coordinate dependent transformation on the hyper-sphere (norm-preserving), we applied a clustering algorithm to the vectors corresponding to nouns in the data set of our model. The intuitive purpose of this procedure was to study the local behavior of the transformation associated to a given adjective, hence its action on specific categories of nouns, such as family members, animals, food etc. Furthermore, this discretization of the surface of the hyper-sphere allowed us to train a vector representation only for the images of the cluster centers under the adjectives, rather than for every single noun vector. In our setting, whenever two vectors corresponding to nouns n1 and n2 belong to the same cluster j, all the occurrences of the bigrams adj+n1, adj+n2 contributed to the representation of the same vector $\mathbf{Adj}[\mathbf{C_j}]$, which is the (approximated) image of the j-th cluster center under the action of the adjective adj .
This method has the main advantage of solving one of the biggest problems when training a representation for bi-grams: the number of occurrences of a specific n-gram in the corpus (sequence of n words or tokens) decreases really steeply with n.
This means that when learning the vector representation of a given adj+noun bigram, our model won't distinguish between "old man" or "old dude", whereas, for example, it would consider "old man" and "old dog" as two separate bi-grams.
We can think about this procedure as a blurred sight on the dictionary with semantic similarity (more precisely with its approximation via vector cosine similarity) as a clustering criteria.

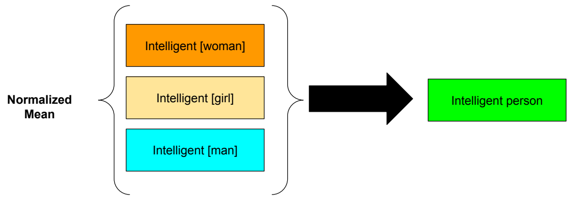


Figure 18: Scheme of the construction of the image vectors using clusters

The word vectors can be clustered on the sphere using different algorithms such as k-means and its variants or DBScan. We opted for the latter since it doesn't require to know the number of clusters we are going to find in advance.
This procedure obviously provides an approximation to the actual transformation, which becomes more and more accurate as we decrease the dimension of the clusters. On the other hand, making the latter too small would make the number of occurrences of the relative bigrams in the corpus drop, affecting the quality of the representation. We therefore assumed that it is possible to find a balance between these two tendencies.

7.3 DBScan Clustering Algorithm and the problem of dishomogenous density

DBScan [Est+96] performs a clusterization of vectors which is based on two parameters, namely
minPoints : the minimum number of points required to form a cluster.
epsilon: the minimum distance between a given point and the closest core point to consider the first one a core point as well.
According to these two parameters vectors are classified as core points or border points of a given cluster or as noise points, which don't belong to any cluster. The algorithm consists in the following steps:

- 1) Find all the neighbor points within eps and identify the core points or visited with more than MinPoints neighbors.
- 2) For each core point if it is not already assigned to a cluster, create a new cluster.
- 3) Find recursively all its density connected points and assign them to the same cluster as the core point.
- 4) A point a and b are said to be density connected if there exist a point c which has a sufficient number of points in its neighbors and both the points a and b are within the eps distance. This is a chaining process. So, if b is neighbor of c, c is neighbor of d, d is neighbor of e, which in turn is neighbor of a implies that b is neighbor of a.
- 5) Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

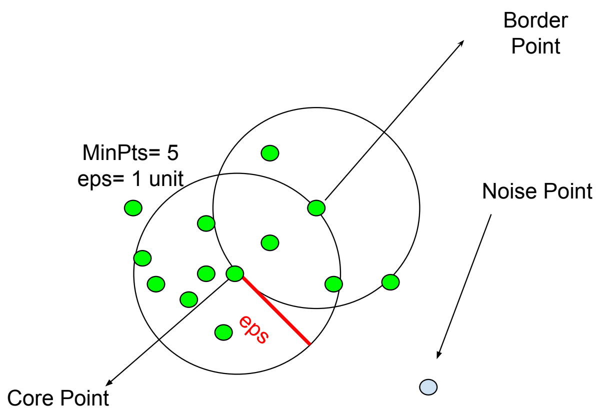


Figure 19: Illustration of DBScan algorithm

Although this algorithm is quite fast and precise, the quality of its performance drops considerably when the density of the points we want to cluster is highly dishomogeneous. Unfortunately, this is often the case when dealing with word vectors of English dictionary, where the distance between word vectors has a wide range.
For example, this is the radial density function of the model ”glove-wiki-gigaword-300”.
Later in this work we will use a different data set: ”word2Vec-google-news-300”.`https://huggingface.co/fse/word2vec-google-news-300`

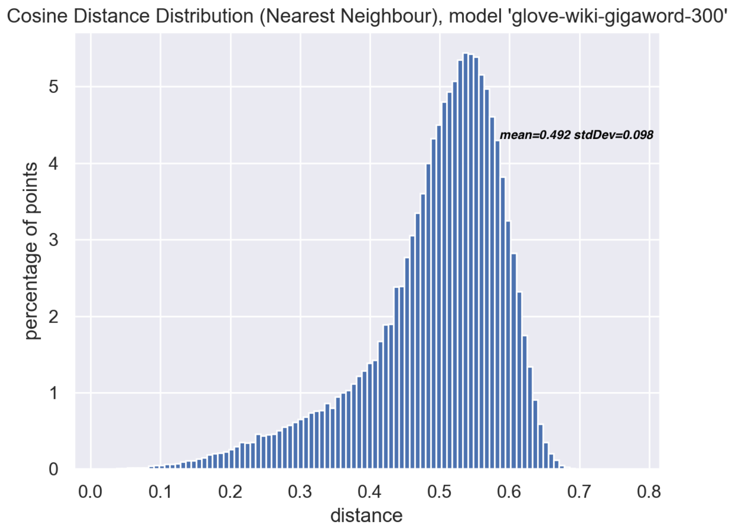


Figure 20: Cosine Distance Distribution of data set glove-gigaword-300.

We would like to compare this plot to the one related to a random distribution of points on a hyper-sphere. Therefore,

we will now provide the limiting value of the distance of two random points whose angles on the hyper-sphere are uniformly randomly sampled as the number of dimensions the goes to infinity.
Following the reasoning of the article [Blu60] starting from the expression of the n-Dimensional Spherical Coordinates:

$$\begin{cases} x_1 = rcos(\phi_1) \\ x_2 = rsin(\phi_1)cos(\phi_2) \\ ... \\ x_{n-1} = rsin(\phi_1)..sin(\phi_{n-2})cos(\theta) \\ x_n = rsin(\phi_1)..sin(\phi_{n-2})sin(\theta) \end{cases} \tag{7}$$

We can prove by induction that the Jacobian is given by the expression

$$J_n = J(r, \theta, \phi_1, \phi_2, ..., \phi_{n-2}) = r^{n-1} \prod_{k=1}^{n-2} sin^{n-1-k}(\phi_k) \tag{8}$$

where $\theta \in [0, 2\pi], r = 1, \phi_k \in [0, \pi] \forall k$
If we now consider two random points on the sphere by symmetry we can consider on of them sitting at one of the hyper-poles with coordinates $x = (0, 0, ...0, 1)$. If we now compute its Euclidean distance with another random point y on the hyper-sphere we get the expression

$$\sqrt{\sum_{k=1}^{n} (x_k - y_k)^2} = \sqrt{sin^2(\phi_1)..sin^2(\phi_{n-2})sin^2(\theta) + sin^2(\phi_1)..sin^2(\phi_{n-2})cos^2(\theta) + .. + sin^2(\phi_1)cos^2(\phi_2) + (1 - cos(\phi_1))^2} \tag{9}$$

$$= \sqrt{sin^2(\phi_1) + cos^2(\phi_1) + 1 - 2cos(\phi_1)} = \sqrt{2(1 - cos(\phi_1))}$$

(10)

We now average this value over the surface of the n-sphere

$$\frac{\int_0^{2\pi} \int_0^{\pi} .. \int_0^{\pi} \sqrt{2(1 - cos(\phi_1))} \prod_{k=1}^{n-2} sin^{n-1-k}(\phi_k)\, d\phi_1 .. d\phi_{n-2}\, d\theta}{\int_0^{2\pi} \int_0^{\pi} .. \int_0^{\pi} \prod_{k=1}^{n-2} sin^{n-1-k}(\phi_k)\, d\phi_1 .. d\phi_{n-2}\, d\theta}$$

(11)

Canceling out the common factors we have

$$= \sqrt{2} \frac{\int_0^{\pi} \sqrt{(1 - cos(\phi_1))} sin^{n-2}(\phi_1)\, d\phi_1}{\int_0^{\pi} sin^{n-2}(y)\, dy}$$

(12)

If we consider an even number of dimensions $n = 2p$ we obtain

$$= \sqrt{2}\frac{\int_0^{\pi} (1 - cos(\phi_1))^{\frac{1}{2}} (1 - cos^2(\phi_1))^{\frac{2p-3}{2}} \, dcos(\phi_1)}{\frac{\pi(2(p-1))!}{2^{2(p-1)}\cdot((p-1)!)^2}}$$

(13)

where we used the relation

$$\int_0^{\frac{\pi}{2}} sin^a(\theta) \cdot cos^b(\theta) \, d\theta = \frac{1}{2}\beta(\frac{a+1}{2}, \frac{b+1}{2}) \tag{14}$$

We perform the change of variable $x = cos(\phi_1)$ and get

$$= \sqrt{2}\frac{\int_{-1}^{1} (1-x)^{p-1}(1+x)^{\frac{2p-3}{2}} \, dx}{\frac{\pi(2(p-1))!}{2^{2(p-1)}\cdot((p-1)!)^2}} \tag{15}$$

The first integral can be performed by parts obtaining the expression

$$\int_{-1}^{1} (1-x)^{p-1}(1+x)^{\frac{2p-3}{2}} \, dx = \frac{2^{3p-\frac{3}{2}}(2p-3)!!(p-1)!}{(4p-3)!!} \tag{16}$$

If we now combine everything we get

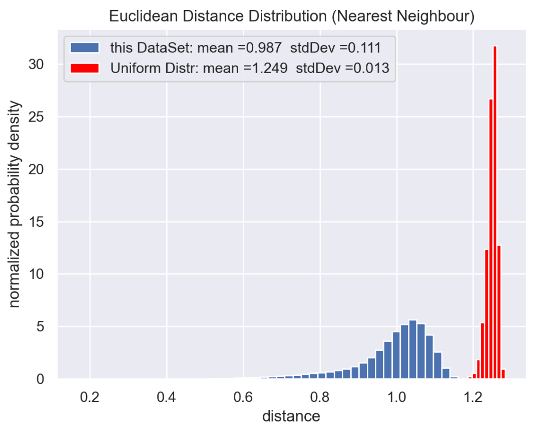
$$< d(x,y) >_{n=2p} = \sqrt{2}\frac{2^{5p-\frac{7}{2}}(2p-3)!!((p-1)!)^3}{\pi(2(p-1))!(4p-3)!!} \tag{17}$$

Making use of Stirling's approximation formulas for large ns

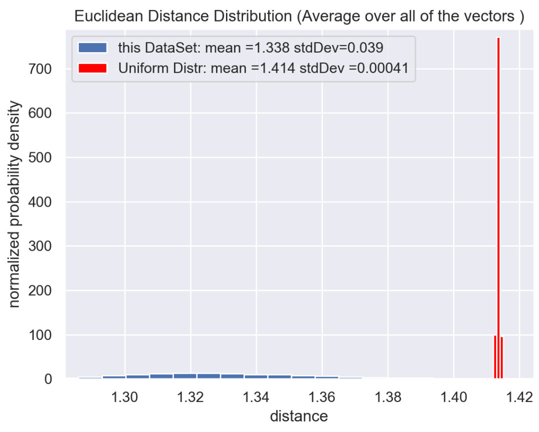
$$\begin{cases} n! \approx (2)^{\frac{1}{2}}(\frac{n}{e})^n \\ (2n+1)!! \approx (2(2n+1))^{\frac{1}{2}}(\frac{n}{e})^{\frac{n}{2}} \end{cases} \tag{18}$$

we get

$$< d(x,y) >_{n=2p} \approx \sqrt{2} \tag{19}$$

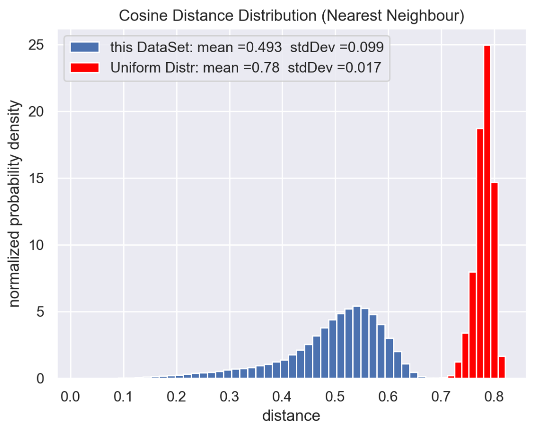


(a) Comparison between the closest neighbor distance distributions of data set and the uniform distribution on S^{n-1}

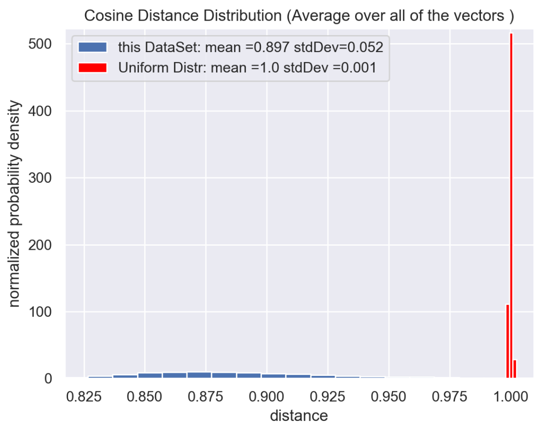


(b) Comparison between the average distance distributions of data set and the uniform distribution on S^{n-1}

Figure 21: Euclidean Distance



(a) Comparison between the closest neighbor distance distributions of data set and the uniform distribution on S^{n-1}



(b) Comparison between the average distance distributions of data set and the uniform distribution on S^{n-1}

Figure 22: Cosine Distance

7.3.1 A possible solution

As a possible solution to the problem of varying density of the points on the hyper-sphere, we tried a zoom-in approach, where the epsilon values starts from its maximum value on the dataset (which is about 0.65) and then decreases gradually by discrete steps. For each epsilon we run the DBScan algorithm for a fixed value of minSample. The

algorithm outputs a certain number of clusters, which are represented by lists of indices. We store the clusters whose cardinality is under a certain threshold and then we recursively run DBScan on the ones that are above it, this time with an epsilon that is reduced by a constant value.
At each step the procedure classifies a certain number of points as noise points, which don't belong to any cluster given the current parameters. A fraction of these rejected points is randomly added back to the input of the biggest cluster we recursively run the algorithm on. In this way DBScan considers increasingly higher densities of the points, starting from a large scale view on the data and then zooming onto the the details of the smaller structures.
For this specific dataset we found that 5 is a good value for minPoints, which means we expect the smaller clusters of the data to have cardinality close to this value. We set the threshold of the recursive process to 50 such that we considered clusters whose cardinality is larger than this value as compound structures. Since we were only interested in noun clustering, we filtered the dictionary using SpaCy python library which is able to classify words according to their grammar types, keeping only the word vectors associated to nouns. We started from a value of epsilon of 0.6 since it is close to the right edge of the support of the closest neighbour distance density. At each recursive step we decrease epsilon by the small value 0.02, this to make sure that we don't miss any important structure of the data. We then save the normalized center vector of each cluster on the hypersphere surface and assign to it a name that is given by the three closest words to the vector. The tree structure of the nested clusters is encoded in the name of their files as an adress of the form "node.node. .. .node". This information allows us to plot a representation of the cluster tree as a pdf file.

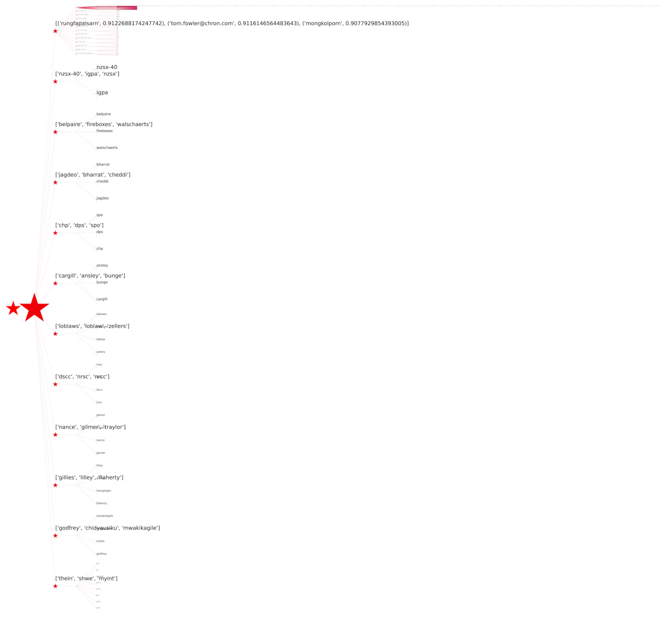
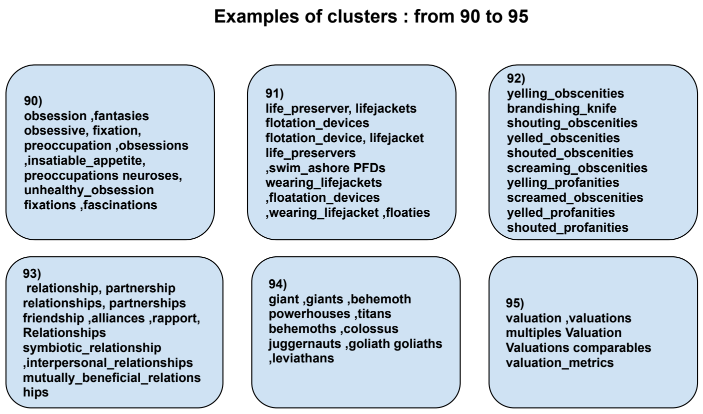

[('rungfapaisarn', 0.9122688174247742), ('tom.fowler@chron.com', 0.9116146564483643), ('mongkolporn', 0.9077929854393005)]
nzsx-40
['nzsx-40', 'igpa', 'nzsx']
igpa
belpaire
['belpaire', 'fireboxes', 'walschaerts']
fireboxes
walschaerts
bharrat
['jagdeo', 'bharrat', 'cheddi']
cheddi
jagdeo
spo
['chp', 'dps', 'spo']
dps
chp
ansley
['cargill', 'ansley', 'bunge']
bunge
cargill
loblaws
['loblaws', 'loblaw', 'zellers']
loblaw
zellers
nrsc
['dscc', 'nrsc', 'nrcc']
dscc
nrcc
gilmer
['nance', 'gilmer', 'traylor']
nance
garner
lilley
['gillies', 'lilley', 'flaherty']
macgregor
flaherty
mwakikagile
['godfrey', 'chidyausiku', 'mwakikagile']
myles
godfrey
['thein', 'shwe', 'myint']

Figure 23: Cluster Tree Plot

7.3.2 Procedure Results

After tuning the hyper-parameters this clustering procedure output 10438 clusters with average Euclidean radius (0.63 ± 0.09) and average number of elements 4.7. Here we show some examples of clusters output by the procedure.


Examples of clusters : from 90 to 95

90)
obsession ,fantasies
obsessive, fixation,
preoccupation ,obsessions
,insatiable_appetite,
preoccupations neuroses,
unhealthy_obsession
fixations ,fascinations

91)
life_preserver, lifejackets
flotation_devices
flotation_device, lifejacket
life_preservers
,swim_ashore PFDs
wearing_lifejackets
,floatation_devices
,wearing_lifejacket ,floaties

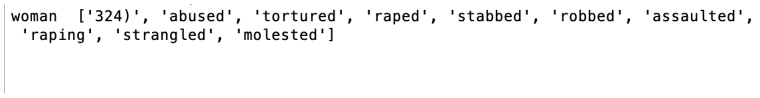
92)
yelling_obscenities
brandishing_knife
shouting_obscenities
yelled_obscenities
shouted_obscenities
screaming_obscenities
yelling_profanities
screamed_obscenities
yelled_profanities
shouted_profanities

93)
relationship, partnership
relationships, partnerships
friendship ,alliances ,rapport,
Relationships
symbiotic_relationship
,interpersonal_relationships
mutually_beneficial_relations
hips

94)
giant ,giants ,behemoth
powerhouses ,titans
behemoths ,colossus
juggernauts ,goliath goliaths
,leviathans

95)
valuation ,valuations
multiples Valuation
Valuations comparables
valuation_metrics

(a) Examples of clusters : from 90 to 95


```
woman  ['324)', 'abused', 'tortured', 'raped', 'stabbed', 'robbed', 'assaulted',
 'raping', 'strangled', 'molested']
```

(b) Example of social bias concerning the word "woman". This cluster containing words related to femicide and gender related harassment appeared in the neighbourhood of the word "woman".
.

To speed up the procedure of finding the closest cluster center to a given word we used the tree search algorithm KDTree from scipy.spatial library. While fine tuning the hyper-parameters we encountered some interesting phenomenas such as cultural biases. In the clustering procedure, in order to preserve the structure of the data set, we didn't filter out the noun embeddings corresponding to adjectives or to the past participle of verbs. As a result, we could observe how certain concepts were linked together by the model. For example here 24b we show how, for a certain choice of the hyper-parameters, the clustering procedure assigns a cluster containing words related to femicide and gender related harassment to a position which belongs to the neighbourhood of the word "woman".

7.4 Using clusters to train adjective representations

As we previously explained, clustering vectors allows us to automatically (and approximately) find out which nouns are synonyms or at least quite similar to each other so that we can consider them as the same word when training the representation of each adjective+noun bigram.
The training procedure consists in the following steps:
0) At first the document is processed, its corpus is divided into sentences and the latter are divided into single words. This words are then filtered in order to remove symbols and punctuation characters.

1) Secondly, the text is scrolled sentence by sentence looking for adjectives. Every time a bigram adjective+noun is found with position respectively i and i+1 inside the sentence, we consider all the words in the same sentence the fall inside the window
Sentence[max(i-windowSize,0),i-1] and Sentence[i+2,min(i+windowSize,len(Sentence)-1)]
and we compute the normalized mean of their vectors. More precisely, we calculated the weighted mean, assigning to each word vector a weight that is equal to the inverse of the logarithm of its frequency in the whole corpus plus one, such that $W_i = \frac{1}{\log_2(f(word_i))+1}$.
This trick allows us to take into account how relevant is the presence of a specific word in the context of our bi-gram according to its relative frequency in the text.

2) For the next step we look for the cluster to which the noun belongs and we add the mean vector to the corresponding list. We obtain a list of lists which has the following structure:

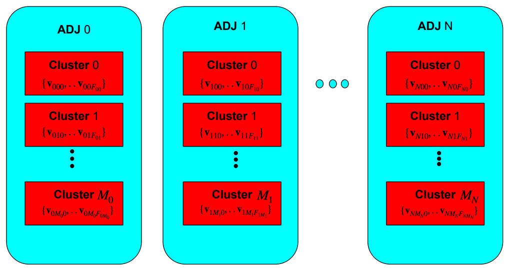

ADJ 0
Cluster 0
$\{\mathbf{v}_{000}, .. \mathbf{v}_{00F_{00}}\}$
Cluster 1
$\{\mathbf{v}_{010}, .. \mathbf{v}_{01F_{01}}\}$
Cluster M_0
$\{\mathbf{v}_{0M_00}, .. \mathbf{v}_{0M_0F_{0M_0}}\}$
ADJ 1
Cluster 0
$\{\mathbf{v}_{100}, .. \mathbf{v}_{10F_{10}}\}$
Cluster 1
$\{\mathbf{v}_{110}, .. \mathbf{v}_{11F_{11}}\}$
Cluster M_1
$\{\mathbf{v}_{1M_10}, .. \mathbf{v}_{1M_1F_{1M_1}}\}$
ADJ N
Cluster 0
$\{\mathbf{v}_{N00}, .. \mathbf{v}_{N0F_{N0}}\}$
Cluster 1
$\{\mathbf{v}_{N10}, .. \mathbf{v}_{N1F_{N1}}\}$
Cluster M_N
$\{\mathbf{v}_{NM_N0}, .. \mathbf{v}_{NM_NF_{NM_N}}\}$

Figure 25: Scheme of the list representing vectors divided in different clusters with respect to different adjectives.

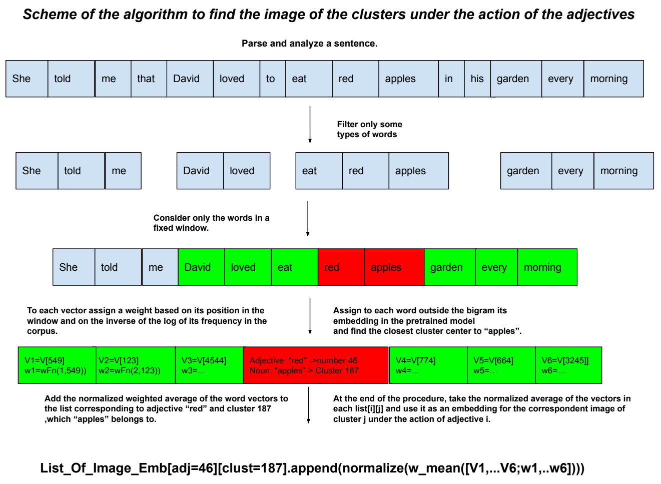

Scheme of the algorithm to find the image of the clusters under the action of the adjectives
Parse and analyze a sentence.
She told me that David loved to eat red apples in his garden every morning
Filter only some types of words
She told me
David loved
eat red apples
garden every morning
Consider only the words in a fixed window.
She told me David loved eat red apples garden every morning
To each vector assign a weight based on its position in the window and on the inverse of the log of its frequency in the corpus.
Assign to each word outside the bigram its embedding in the pretrained model and find the closest cluster center to “apples”.
V1=V[549] w1=wFn(1,549))
V2=V[123] w2=wFn(2,123))
V3=V[4544] w3=…
Adjective: “red” ->number 46 Noun: “apples”-> Cluster 187
V4=V[774] w4=…
V5=V[664] w5=…
V6=V[3245]] w6=…
Add the normalized weighted average of the word vectors to the list corresponding to adjective “red” and cluster 187 ,which “apples” belongs to.
At the end of the procedure, take the normalized average of the vectors in each list[i][j] and use it as an embedding for the correspondent image of cluster j under the action of adjective i.
List_Of_Image_Emb[adj=46][clust=187].append(normalize(w_mean([V1,...V6;w1,..w6])))

Figure 26: Scheme of the algorithm we used to build a representation of the image vectors

3)The third step consists in taking the normalized mean of all the center vectors in each cluster and considering it as the center of the new cluster transformed by the adjective. Since we possess a list containing all the adjectives and another one which contains all the clusters, we will refer to the centers of the transformed clusters as V_{ij} where i indicates the adjective and j the specific cluster. In the scheme in picture 25 this correspond to taking the normalized average of all the vectors belonging to a specific cluster j which corresponds to a certain adjective i. In the following chapters, when the adjective is kept fixed, we will simply refer to them as $\mathbf{y_{CM,i}}$.

7.5 The problem of biases in the data set

The technique we used allowed us to build the vectorial representation of a new word, in our case the bigram adj+nounCluster, starting from an existing pre-trained model like ”word2vec-google-news-300”. This method is called ”A la carte Embedding”. As explained by the article [Kho+18], this procedure has important advantages and some disadvantages. For example, it allows to derive a faithful embedding of rare words, for which it is often difficult to collect enough data, from a set of high-quality pre-trained vectors. One of the main limitations consists in the presence of biases, which comes from the fact that the set of all word vectors is seen to share a few common directions.
The ”A la carte ” technique tries to approximate the right feature vector of a bigram by means of an additive feature vector trained on the corpus.
The additive vector of a word w is defined as

$$u_w^{additive} = \frac{1}{|C_W|} \sum_{c \in C_w} \frac{1}{|c|} \sum_{w' \in c} v_{w'} \tag{20}$$

where C_w is the class of all the sets of context vectors where the word w appears in the corpus, c represents a single context set and $v_{w'}$ iterates over all the vectors in the context c. Ideally for every word w which is already present in the model we would like to have

$$u_w^{additive} \approx v_w \tag{21}$$

Since the context embeddings of each bigram are obtained from the mean of all the context vectors, which in turn are defined as the normalized mean of the embeddings of the words within each context window, this main global directions can produced biased context vectors. Simple addition of vectors tends to amplify the component in these directions while suppressing all others that presumably carry important information. A possible remedy is to erase the top three or four component of the vectors when taking the average. This components are derived using the Principal Component Analysis (PCA), which is a algebraic techniques used to obtain an ordered orthonormal set of vectors, the principal components, along which the variance of dataset points is decreasingly maximized. In other words, the first principal component represents the direction along which the variance of the data points is maximized, the second one plays the same role but with the constraint of being orthogonal to the first component we have already fixed and so on. As a first attempt to fix the issue of biases in the data set, we followed the procedure described in paper [JP18] by Jiaqi Mu and Pramod Viswanath.
First of all, they define a partition function for the word vectors as

$$Z(\mathbf{c}) = \sum_{w \in V} e^{\mathbf{c} \cdot \mathbf{v_w}} \tag{22}$$

which can be approximated to second order by the expression

$$Z(\mathbf{c}) \approx |V| + \mathbf{1_{|V|}}^T \cdot W \cdot \mathbf{c} + \frac{1}{2}\mathbf{c}^T \cdot W^T W \cdot \mathbf{c} \tag{23}$$

where W is the matrix with word vectors as rows and $\mathbf{1_{|V|}}$ is a vector of length $|V|$ whose entries are ones.
The authors also define a measure of isotropy as

$$I(V) = \frac{\underset{\|c_w\|=1, w \in V}{min} Z(\mathbf{c_w})}{\underset{\|c_w\|=1, w \in V}{max} Z(\mathbf{c_w})} \tag{24}$$

which in turn can be approximated using (19) as

$$I(V) \approx \frac{|V| + \underset{\|c_w\|=1, w \in V}{min} \mathbf{1_{|V|}}^T \cdot W \cdot \mathbf{c} + \underset{\|c_w\|=1, w \in V}{min} \frac{1}{2}\mathbf{c}^T \cdot W^T W \cdot \mathbf{c}}{|V| + \underset{\|c_w\|=1, w \in V}{max} \mathbf{1_{|V|}}^T \cdot W \cdot \mathbf{c} + \underset{\|c_w\|=1, w \in V}{max} \frac{1}{2}\mathbf{c}^T \cdot W^T W \cdot \mathbf{c}} = \frac{|V| - \|1_{|V|}^T W\| + \frac{1}{2}\sigma_{min}^2}{|V| + \|1_{|V|}^T W\| + \frac{1}{2}\sigma_{max}^2} \tag{25}$$

where σ_{min} and σ_{max} are respectively the smallest and the largest singular values of W whereas their squared values correspond to the smallest and the largest eigenvalues of $W^T W$.
As the vectors become more and more isotropic, which means that they lose any preferential direction, the value of the isotropy I tends to 1. If we impose $I = 1$ in the approximation (21) we get the constraints

$$\begin{cases} \sum_{w \in V} \mathbf{v_w} = 0 \\ \sigma_{min} = \sigma_{max} \end{cases} \tag{26}$$

We therefore should subtract from each vector the mean of their set together with their projection along some of the principal components. Finally, we subtract again the mean from the result. This procedure flattens the spectrum of W and therefore $\sigma_{min} \approx \sigma_{max}$.

For this results we used a subset of different set of pre-trained word vectors called ”word2vec-googleNews-300”, which is known to perform better in the task of word comparison without context. We selected the first 500k vectors in the data set, which are also associated to the most common 500k words in the dictionary including composed words like ”hot dog”.
Firstly, we show how the approximation of Z seems to be good over 1000 vectors randomly picked from the data set

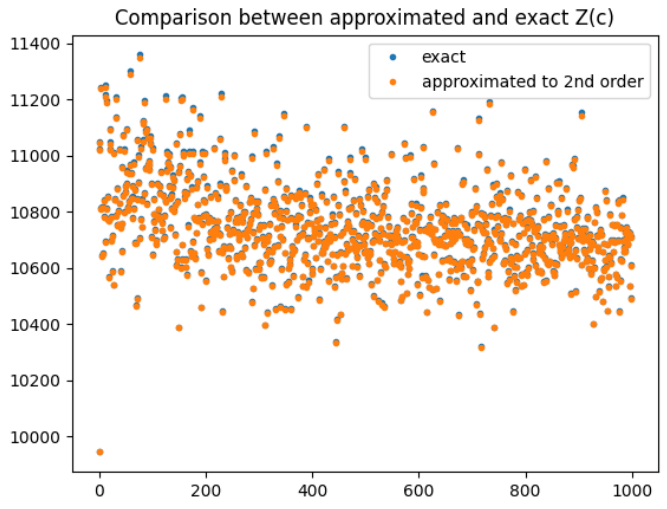


Figure 27: Difference between the exact Z function and its approximation on a set of random vectors

We can then calculate the singular values using Dask library for Pyhton, which uses QR decomposition to outrun the more famous python library Numpy when dealing with strongly rectangular matrices ($dim1 \ll dim2$).
After subtracting their average, the singular values are

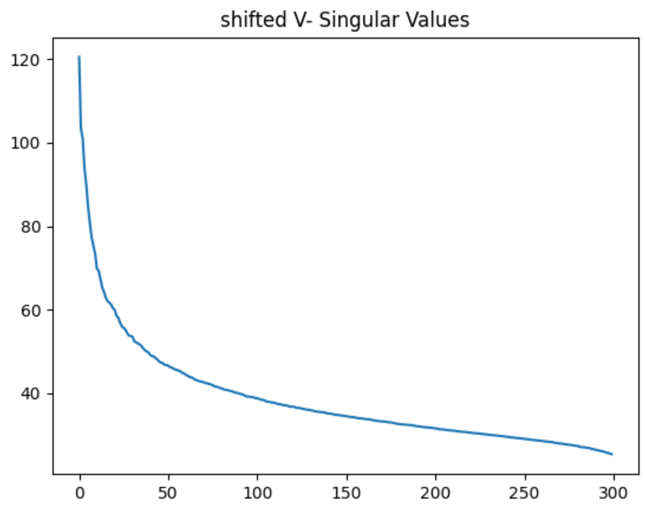


Figure 28: Plot of the singular values of our data set.

The initial approximated value for the isotropy on the original data set is around 0.548, after subtracting the mean vector it raises to 0.979 and it finally reaches 0.986 after removing three principal components.
In order to check our results with the one in paper [JP18], we compare the projections along the first two principal components and the relative frequency of the word in the corpus defined as

$$p(w) = \frac{freq(w)}{\underset{u \in V}{max} freq(u)} \tag{27}$$

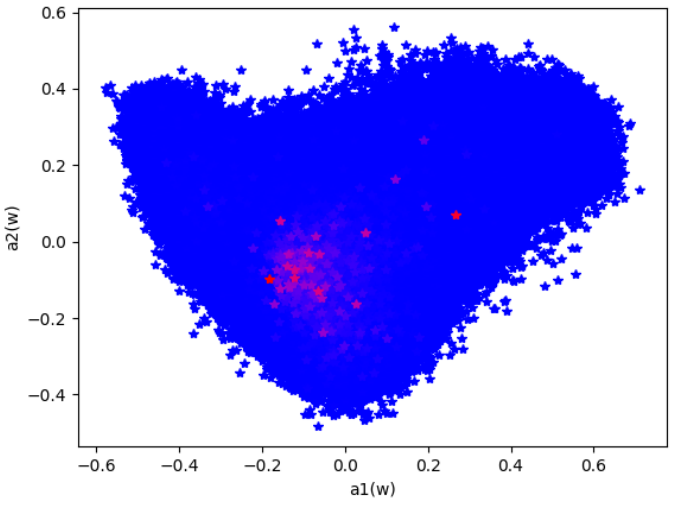


Figure 29: The plot shows the relation between the first two principal components of the word embeddings in our pre-processed data set word2Vec-google-news-300. The color from blue (low) to red (high) indicates the value of $\frac{p(w)}{\underset{w}{max}[p(w)]}$ for the specific word w.

We can see how the most frequent vectors have generally smaller projections along those components, in agreement with what found be the authors of the paper.

7.5.1 Norm of Vectors

We also plotted the distribution of the norms of these vectors before normalization.

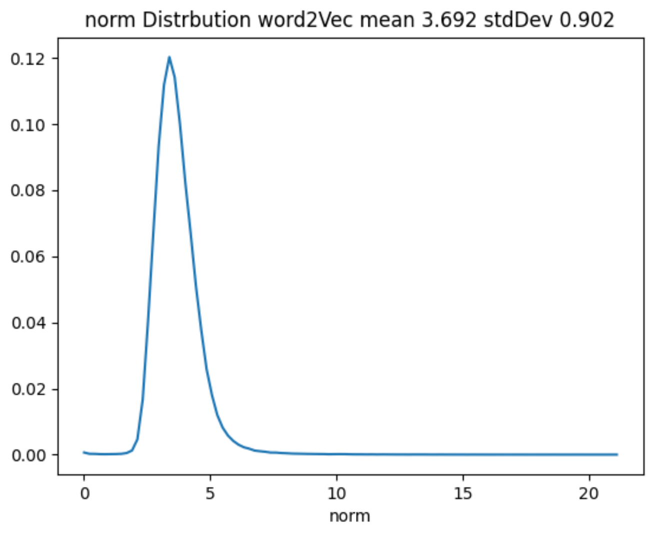
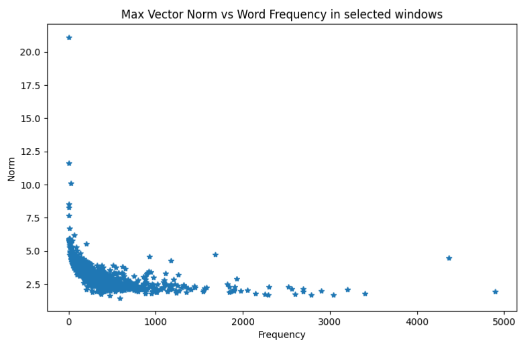
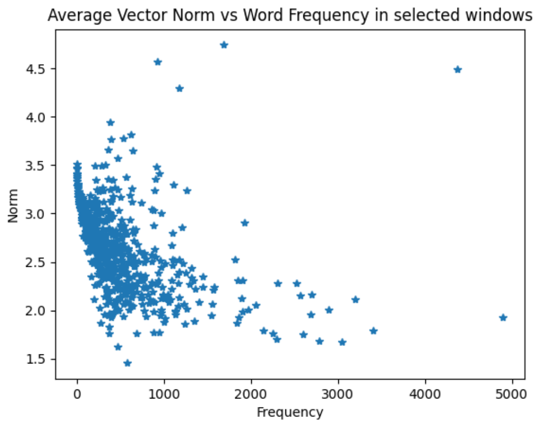


Figure 30: Distribution of the norms of the word embeddings in the data set "word2vec-google-news-300"

In the paper "Measuring Word Significance using Distributed Representations of Words" [SW15] by Shakel and Wilson, the authors mention how, although they are usually ignored when computing word similarity or related tasks, the norms of the vectors carry information about the distribution of the correspondent words in the corpus.
Here we plot their norm vs their frequency in the corpus we used for the training of the representation of the adjectives. Whenever two words had the same frequency we took the mean or the max of the norms of the correspondent vectors.





(a) Max of the norms of the word embeddings vs the frequency of the relative word

(b) Average of the norms of the word embeddings vs the frequency of the relative word

7.6 Results of the procedure

In the rest of the paper the reference data set of word vectors will be this more isotropic version of the previously mentioned ”word2vec-google-news-300-top500k”. As we found out, this procedure visibly improved the quality of the transformed cluster vectors, which previously often converged to weird location on the hyper-sphere corresponding to words completely unrelated to considered the adj+noun pair.

7.7 Further improvements

As showed in the paper [Kho+18], this procedure is not the most efficient and is likely to lose an important part of the ”signal”. Khodak and Saunshi et al proposed a different approach, which modifies the additive vectors by the action of a linear transformation

$$\mathbf{u}_{\mathbf{w}}^{\mathbf{additive}} = \frac{1}{|C_W|} \sum_{c \in C_w} \sum_{w' \in c} \mathbf{v}_{\mathbf{w'}} \tag{28}$$

where we assumed the length of the context window $|c|$ to be constant. This linear transformation is learned from the corpus, in particular it is chosen to minimize the distance between the word embeddings of already known words ($\in$ vocabulary V) and their additive approximation

$$A = \underset{A' \in M(R)^{mxm}}{argMin} \sum_{w \in V} \|\mathbf{v}_{\mathbf{w}} - A' \mathbf{u}_{\mathbf{w}}^{\mathbf{additive}}\|^2 \tag{29}$$

Once we have learnt this transformation, we can apply it to any additive approximation of a new n-gram in order to improve its quality and therefore go from a context to a feature vector

$$\mathbf{v}_{\mathbf{f}} = A \mathbf{u}_{\mathbf{f}} \tag{30}$$

7.8 Adapting this approach to rotations

7.8.1 The initial approach: Lie Groups

In our case, since we are dealing with vectors which belong to an hyper-sphere, it seems logical to work with rotation matrices. Later in our discussion we will show that this is reasonable at least locally. The problem we want to solve is hence finding the rotation that best transforms our approximating additive vectors into the already existing high quality ones from the model. Once we posses this matrix we can let it act on the additive vectors of new words in order to maximize their quality and coherence with respect to the model. This operation can be summarized as

$$R = \underset{R' \in M(R)^{dxd}}{argMin} \sum_{w \in V} \|\mathbf{v}_{\mathbf{w}} - R' \mathbf{u}_{\mathbf{w}}^{\mathbf{additive}}\|^2 \tag{31}$$

Since rotations form a Lie group [Hsi17] we can instead work with the group angles as free real parameters over which we minimize

$$\begin{cases} R = e^{\sum_{i=1}^{m} \theta_i A_i} \\ m = \frac{d(d-1)}{2} \\ A_i \in AntiSym(d), \forall i \end{cases} \tag{32}$$

where the matrices A are the anti-symmetric generators of SO(d).
In order to minimize the cost function defined above we can use a stochastic gradient descent algorithm. This procedure computes the gradient of the cost function at each step while only considering a randomly picked subset of the word vectors v_w involved in the sum. It then updates the values of the angles moving in the parameters space by a fixed step in the opposite direction of the gradient in that point, so

$$\begin{cases} C = \sum_{w \in V} \|\mathbf{v_w} - R'\mathbf{u_w}^{additive}\|^2 \\ \nabla_i C(\theta) = \sum_{w \in V} (\mathbf{v_w} - R\mathbf{u_w}^{additive})(\nabla_i R\mathbf{u_w}) \\ \theta' = \theta - dx\nabla C(\theta) \end{cases} \tag{33}$$

The object ∇R is a stack of matrices which represents the derivative of the rotation matrix in d dimensions at a given point. In order to compute such a gradient we explored different alternatives, namely
1) Approximating the matrix exponential using its definition

$$e^A = \sum_{i=0}^{\infty} \frac{A^n}{n!} \tag{34}$$

2) Using Duhamel's formula for the derivative of the exponential map

$$\frac{d}{dt}(e^{A(t)}) = e^{A(t)} \frac{I - e^{ad_A}}{ad_A} \frac{dA(t)}{dt} \tag{35}$$

where $ad_A[X] = [A, X]$ is the adjoint action of a Lie algebra on itself [Hsi17].

3) Or applying Lie's formula for the exponential of a sum of matrices

$$\frac{d}{dt} e^{A+B} = \lim_{n \to \infty} (e^{\frac{A}{n}} e^{\frac{B}{n}})^n \tag{36}$$

where in our case $A = \sum_{i=1}^{m} \theta_i A_i$ and the A_is do not commute. As our matrix is defined as a sum of more than 40k matrices the first approach looked less computational appealing as we would have to compute the number of products at each step j of the Taylor expansion would grow as $(\frac{d(d-1)}{2})^j$. We therefore explored the second and the third options.
2) the fraction in Duhamel's formula can be computed as

$$\frac{I - e^{ad_A}}{ad_A} = \sum_{k=0}^{\infty} \frac{-1^k}{(k+1)!} (ad_A)^k \tag{37}$$

In our case

$$\frac{d}{d\theta_i}(e^A) = e^{A(t)} \sum_{k=0}^{\infty} \frac{-1^k}{(k+1)!} (ad_A)^k [A_i] \tag{38}$$

where

$$(ad_A)^k [A_i] = [A, [A, \underbrace{[...[A, A_i]]...]}_{ktimes} \tag{39}$$

2) For what concerns Lie Formula approach we have

$$\frac{d}{d\theta_l}(e^A) = \frac{d}{d\theta_i}\lim_{n\to\infty}(\prod_{i=1}^{m} e^{\frac{\theta_i}{n}A_i})^n = \tag{40}$$

$$\lim_{n\to\infty}\frac{d}{d\theta_l}(\prod_{i=1}^{m} e^{\frac{\theta_i}{n}A_i})^n = \tag{41}$$

Since the order of the factors does not matter in Lie Formula, we can rearrange the product and use the formula for the derivative of a matrix power to write

$$\lim_{n\to\infty}\sum_{i=0}^{n-1}[(\prod_{i=0}^{m} e^{\frac{\theta_i}{n}A_i})^i \cdot \frac{d}{d\theta_l}(e^{\frac{\theta_l}{n}A_l} \cdot \prod_{i\neq l}^{m} e^{\frac{\theta_i}{n}A_i}) \cdot (\prod_{i=0}^{m} e^{\frac{\theta_i}{n}A_i})^{n-i}] = \tag{42}$$

$$\lim_{n\to\infty}\sum_{i=0}^{n-1}\frac{1}{n}[(\prod_{i=0}^{m} e^{\frac{\theta_i}{n}A_i})^i \cdot A_l(e^{\frac{\theta_l}{n}A_l} \cdot \prod_{i\neq l}^{m} e^{\frac{\theta_i}{n}A_i}) \cdot (\prod_{i=0}^{m} e^{\frac{\theta_i}{n}A_i})^{n-i-1}] = \tag{43}$$

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=0}^{n-1}[(e^A)^i A_l (e^A)^{n-i}] \tag{44}$$

This method seems to converge quite fast, providing a good approximation even for n=6.
Furthermore, this procedure allows us to compute the array of matrix containing powers of e^A from 0 to n una tantum and reuse it for each component of the gradient.

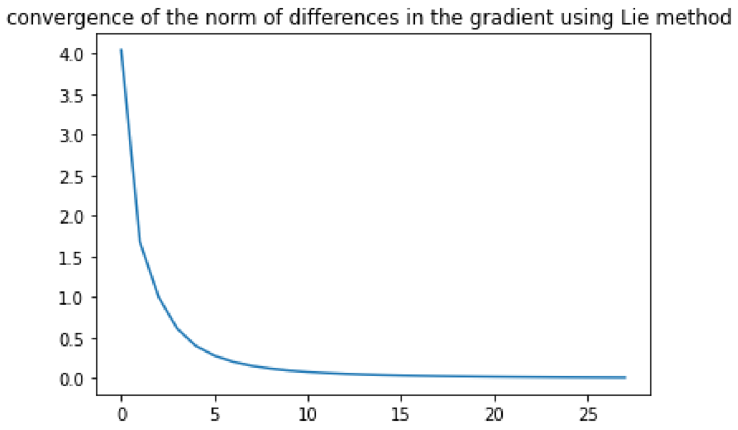


Figure 32: Norm of the differences between each step of the approximation using Lie method to approximate the matrix gradient $\frac{d}{d\theta_l}(e^A)$

7.8.2 A help from the literature: The Orthogonal Procrustes Problem

The problem of finding the orthogonal matrix that most closely maps a given matrix A to another matrix B is called Orthogonal Procrustes Problem and is well studied in the scientific literature. The special case where we restrict our matrix to the Special Orthogonal Group of the rotations involves a famous method called Kabsch Algorithm, named after Wolfgang Kabsch [Kab76]. This procedure is useful in graphics as well as in cheminformatics and bioinformatics, where it is applied to compare complex protein structures [Ger91].

7.9 Incorporating Transformers into the procedure

An alternative way to build the vector representation of the transformed cluster is to use the ”Fill-Mask” pipeline of the transformer models. The classifier uses a model called ”distilroberta-base” [San+19]. Every time we encounter a bigram adj+noun, noun $\in cluster_i$, instead of taking the weighted average of the word vectors in our window, we put a mask on the bigram and ask the model to guess its vector representation based on the words in the window (which this time could be as long as multiple sentences).

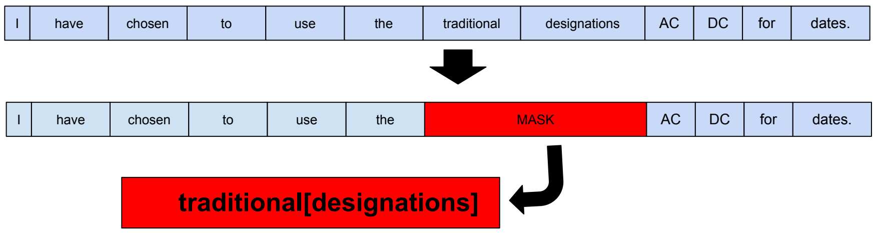
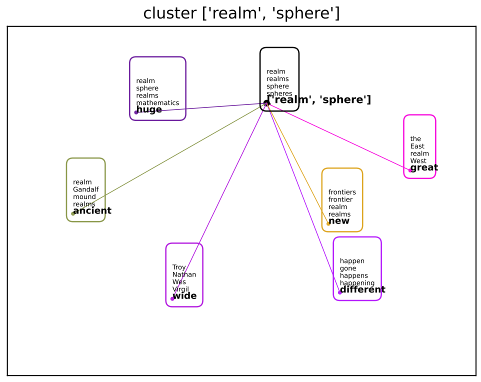
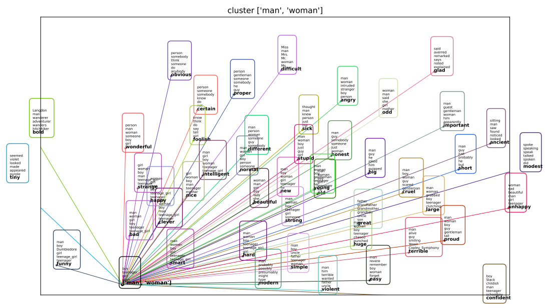


Figure 33: Illustration of the training procedure which incorporates the ”fill-mask” method.

At the end of the training procedure we still average all this predictions for every single cluster and normalize them. This approach turned out to provide considerably more reasonable representations than the previous one .
Here we show some examples of how different the adjectives modify the position of the cluster centers. In order to visually represent the relation between the old cluster center and its images under different adjective transformations we used a technique called Multi Dimensional Scaling or MDS, which we will introduce in section 7.11.2.



(a) MDS of the vectors representing the cluster center of [”realm”,”sphere”,..] and its images under several adjectives. The words in each box correspond to the 4 closest word emebeddings on the n-sphere to the relative image point. The relative distances between the points optimally reflect in 2D the original ones on the n-sphere.



(b) MDS of the vectors representing the cluster center of [”man”,”woman”,..] and its images under several adjectives. The words in each box correspond to the 6 closest word emebeddings on the n-sphere to the relative image point.

7.10 Region of Action of Adjectives

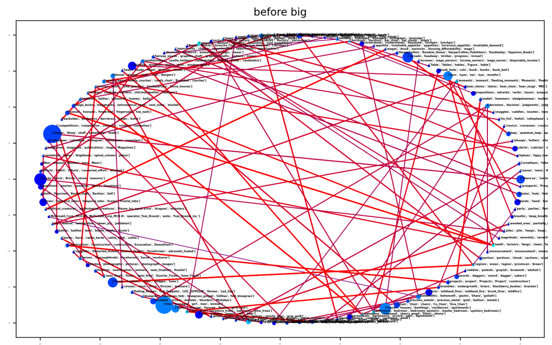
Every given adjective in the English dictionary can act on a certain kind of words, in our case on a specific number of clusters. We call the union of this clusters the ”Region of Action”. For example, the adjective ”old” can act on words concerning living beings like ”man”, ”woman”, ”dog” but also abstract concepts like ”idea”, ”habits” or material things like ”building”. On the contrary, and adjective like ”tall” is not likely to precede abstract words like ”beauty” or ”war”. Accordingly, we expect the region of action of ”old” to be significantly larger and more diverse than the one of ”tall”. Obviously the dimension and the structure of this regions depends on the model and on the particular corpus on which the latter is trained, but we assume as an hypothesis that for a really large and diverse training corpus these regions assume a limiting structure that may only depend on the particular language.
For a given adjective, if we consider all the clusters in its region of action and we calculate the position of their centers on the surface of the hyper-sphere, we end up with a discrete distribution of points which represents the signature of the adjective. More precisely, if we consider their center vectors $\mathbf{x_{CMi}}$ together with the correspondent occurrence frequencies f_i, we can normalize the latter and define its ”activity distribution” . Once we have obtained also a set of image centers under the action of the specific adjective $\mathbf{y_{CMi}}$, using the procedure described in the previous section, we also have information about how this discrete distribution on the n-sphere is changed by the adjective.

7.11 Visualizing the action of the adjectives on the cluster centers

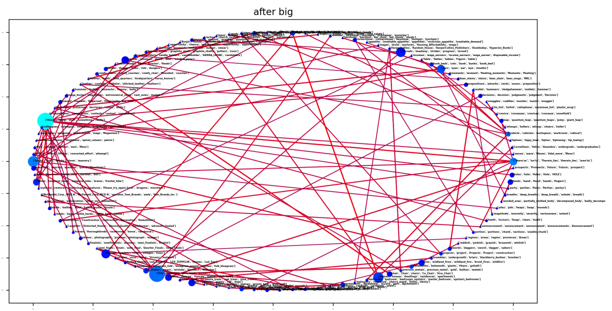
Once we have collected the sets of vectors representing the centers of the the clusters and their image under the action of a specific adjective, it is possible to show how the latter modifies the distances between the clusters.

7.11.1 Graph Representation

Since we are dealing with high dimensional spaces, one option is to use a graph with the cluster centers as nodes and where the thickness of the edges and their color is proportional to the distance between them. The size of the nodes is proportional to the frequency of activity of the adjective on the associated cluster while their color becomes brighter the closer they get to the other cluster centers.



(a) Graph description of the activity distribution of the adjective "big" before its action



(b) Graph description of the activity distribution of the adjective "big" before its action

7.11.2 Multi-dimensional Scaling (MDS)

Another quite powerful and popular option is to use multidimensional scaling, a procedure that allows us to project our n-dimensional points on a plane in such a way that their relative distances are optimally preserved. In other words this 2D plot minimizes the stress function, defined as:

$$S(z) = \sum_i \sum_j (d_{ij} - \|\mathbf{z_i} - \mathbf{z_j}\|)^2 \tag{45}$$

where d_{ij} is a certain distance between the original n-dimensional points $\mathbf{x_i}$ and $\mathbf{x_j}$ and the $\mathbf{z}$ are 2D or 3D vectors. If we apply the MDS algorithm from the python library sklearn to the 500 cluster centers with the biggest radii we get the following plot

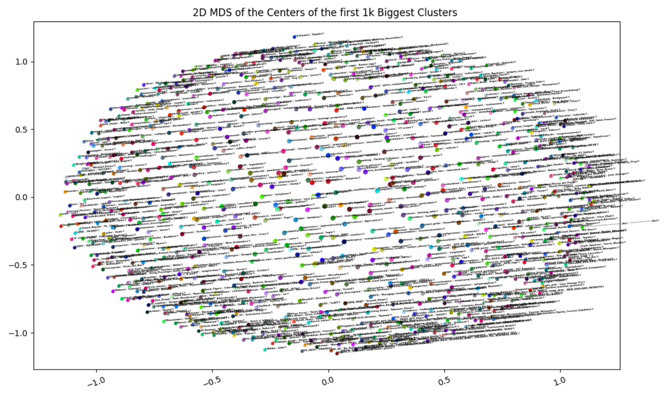
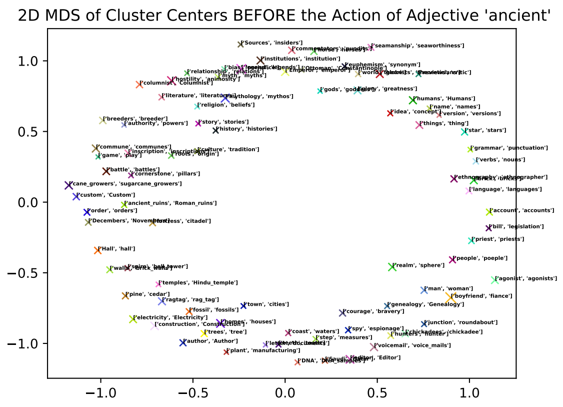
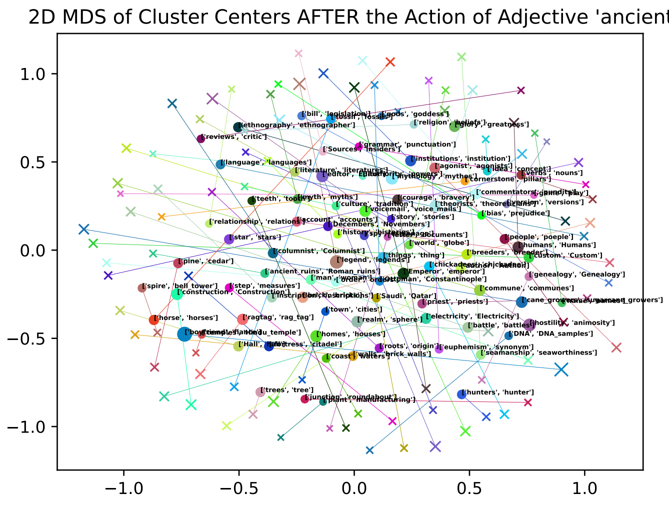


Figure 36: MDS of the 500 cluster centers with the biggest radii

Here the size of the marker correspondent to a given center is inversely proportional to the mean distance from the other centers.
For what concerns the action of the adjectives, here we show the changes in the MDS plot associated to ”ancient”. The arrows show the displacement vectors from the old positions, which are indicated by a ”x” marker.



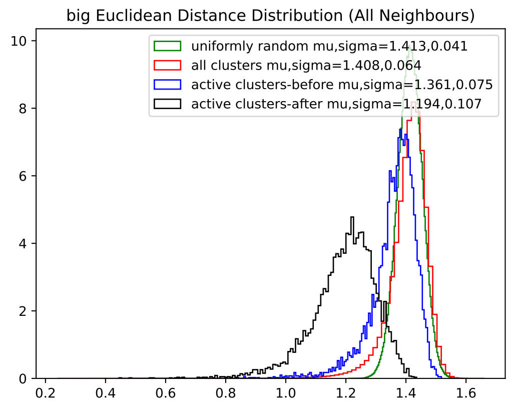
(a) MDS of the activity distribution of the adjective ”ancient” before its action.”



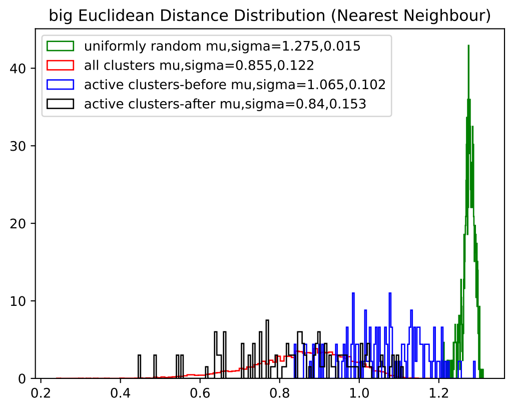
(b) MDS of the activity distribution of the adjective ”ancient” after its action.”

7.11.3 Distance Distribution

Some information can also be gathered from the plot of the distance distribution.



(a) Distance Distribution (All Neighbours)

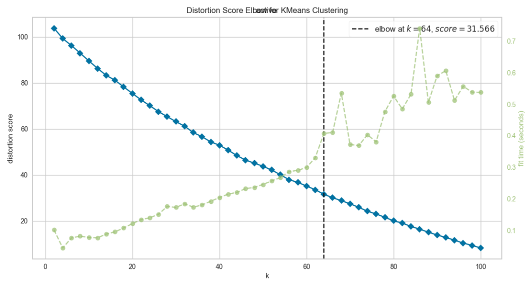


(b) Distance Distribution (Nearest Neighbour)

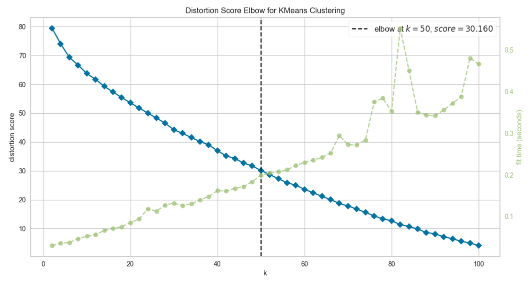
First of all, we notice the the adjectives generally tend to move all the clusters in their region of action closer to each other. The peak in correspondence of square root of two for the uniformly distributed vectors is in perfect agreement with our previous prediction, whereas the fact that the distribution of the nearest neighbor distances of the image clusters has peaks closer to zero supports the idea that at least the action of ”big” presents some condensation points.

7.11.4 K-Means

Since the method described by the author is not easy to efficiently implement on large data sets, for any given adjective adj, we decided to use the faster and more common algorithm k-Means. We performed a clustering of the vectors representing the subset of cluster centers which are active under adj. The algorithm k-means was applied to both the original active cluster centers, before the action of the adjective, and their images under its tranformation, in order to compare the distribution of these vectors on the n-sphere. In particular, we performed a k-elbow analysis, a procedure that helps to determine the optimal number of centroids.



(a) K-elbow test on the active cluster centers of ”Big” before its action



(b) K-elbow test on the active cluster centers of ”Big” after its action

As expected, the action of the adjectives tends to group the vectors, reducing the number of centroids.
The results of the K-Means algorithm (provided by the python libraries Sklearn and yellowbrick) are strongly dependent on the random initial conditions when applied to this data set, we hence tried to double check this result by writing a code in C++ for the spherical k-means algorithm. In the C++ simulation, for which we knew exactly what the code was doing at each step, we see how, although noisy, the elbow test clearly displays a lower optimal value for the number of clusters of the image cluster centers.

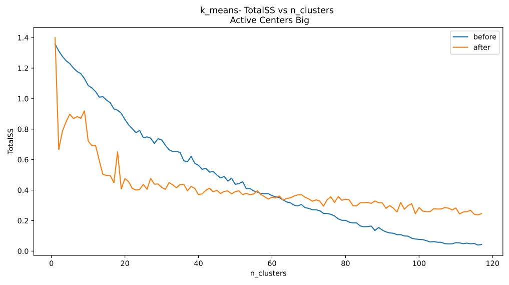


Figure 40: Picture of the k-Means Elbow test performed using our C++ code in order to find the optimal number of clusters for both the original and the transformed cluster centers of the adjective ”big”.

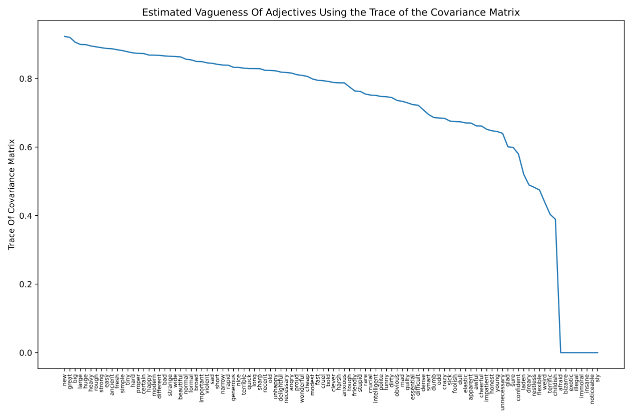
7.11.5 PCA Analysis and Vagueness of an Adjective

A less noisy and more stable procedure could be the Principal Component Analysis of the active cluster centers and their images. Here we show how taking the trace of the pseudo-covariance matrix of the active cluster centers might provide us with an estimation of the vagueness of an adjective, meaning the variety of words on which it is active [Pag84]. Furthermore, comparing this quantity with the same function calculated on the transformed vectors can give us information about the transformation on the variance of data associated to the specific adjective.
The pseudo-covariance matrix is calculated by taking the variance of the vector distribution on the hyper-sphere with respect to their mean, following the expression

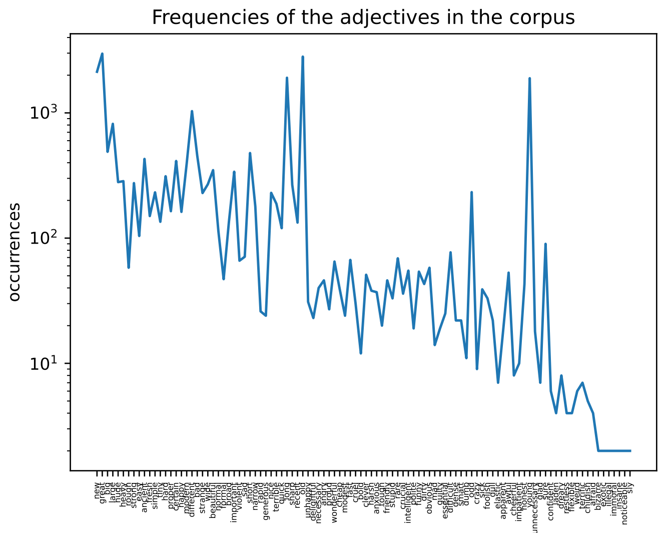
$$J_{ij} = E[(\mathbf{x_i} - \mu)(\mathbf{x_i} - \mu)^T] \tag{46}$$

Although the trace of this matrix doesn't directly depend on the frequency of an adjective in the corpus, since the occurrences of an adjective pair on every specific cluster are normalized and used as probabilities, if an adjective occurs only a few times it is possible to miss some categories of words on which it may be active. This is likely to happen especially when dealing with corpuses with a relatively limited size like the one we used. On the other hand, if an adjective is frequently used, it is more likely that we will collect data about its whole region of action even after a few paragraphs. Another source of positive correlation between frequency and vagueness may come from the fact that by this method, more than directly measuring the vagueness, describes the diversity of word categories on which a given adjective is active. It is therefore reasonable for this property to be closely related to the frequency of occurrence and vagueness of the word. In order to evaluate the correlation, we plot the logarithm of the frequency distribution of the adjectives in our training corpus.

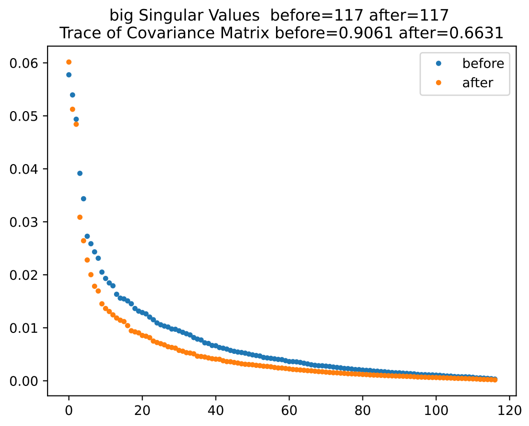
According to the model, the most vague adjectives are ”new”, ”great”, ”big”, ”large”, together with ”easy” and ”strong”. This prediction seems quite reasonable since all of these adjectives are classified as vague gradable adjectives by Semantic Theory, for example [Yas15].



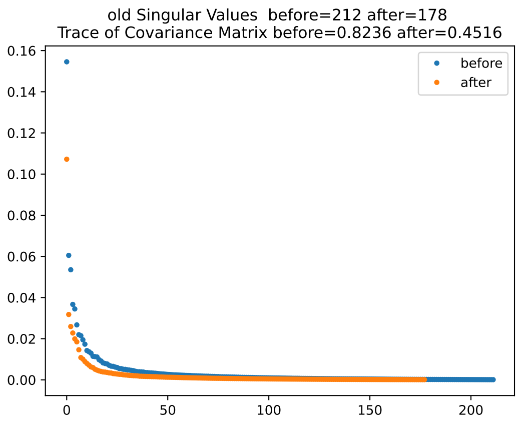
(a) Plot of the estimated vagueness of the adjectives using the trace of the pseudo-covariance matrix calculated on the original cluster centers.



(b) Frequencies of the adjectives in the corpus used for the training phase. Their order reflects the estimated vagueness.



(a) Singular Values of the activity distribution of the adjective ”big” before and after its action on the cluster centers.



(b) Singular Values of the activity distribution of the adjective ”old” before and after its action on the cluster centers.

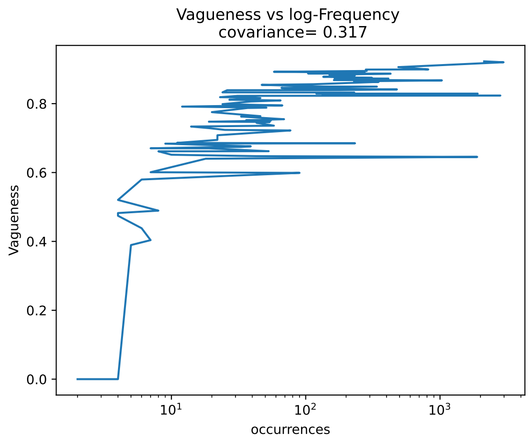


Figure 42: Correlation between the logarithm of the occurrences of an adjective in the corpus and its estimated vagueness.

The following graphs show the change in the singular value decomposition due to the action of the adjectives ”big” and ”old”.

Again we notice how the variance of the data points tends to be reduced along every principal component by the action of the adjective, with a decrease in the dimension of the image space in certain cases.
Looking again at the trace of the covariance matrix we see that the previous order changes significantly:

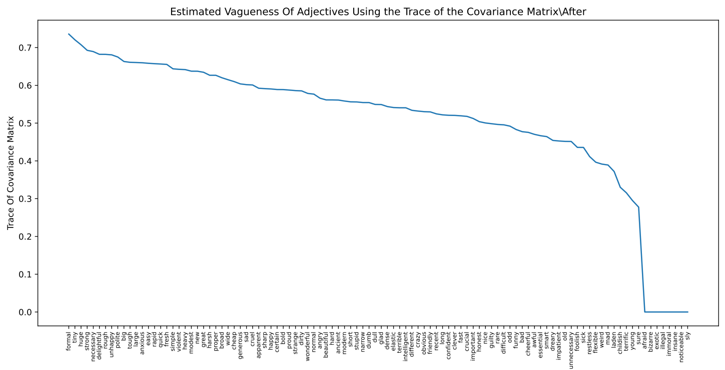


Figure 44: Estimated vagueness of the adjectives in the corpus based on the trace of the covariance matrix of their image cluster centers

7.12 Local Linear approximation as a possible approach

7.12.1 Internal structure of the noun clusters

The method we previously described in section 7.4 helps us to find the center of mass of each transformed cluster under the action of any given adjective. However, examining the clusters we see that they are composed of an average of 5 nouns with an Euclidean radius of about 0.6, which corresponds to an angle of about 35°. The procedure has left us, for each given adjective, with its activity distribution, which consists in a set of pairs frequency-cluster center $(f_i, \mathbf{X_{CMi}})$ and their relative images $\mathbf{Y_{CMi}})$. By construction, for a given adjective we only possess the images of the cluster centers that are active under its action, that are also the ones for which $f_i \neq 0$. This information can help us to approximately reconstruct the adjective map on the whole surface of the n-sphere. In the following sections we will introduce a possible example of a non-linear norm preserving transformation in 3D, as well as explaining some attempts to interpolate the transformation on every point of the n-spherical surface.

7.12.2 A candidate for the transformation in 3D

Since the transformation is norm preserving, we can see it as an angle transformation on the hyper-sphere. In three dimensions, adding the constraint of reducing the average angular distance of points, a possible candidate could be the map

$$\begin{cases} \alpha = \theta(1-k) + \pi k \\ \beta = \phi \end{cases} \tag{47}$$

with $k \in [0, 1]$

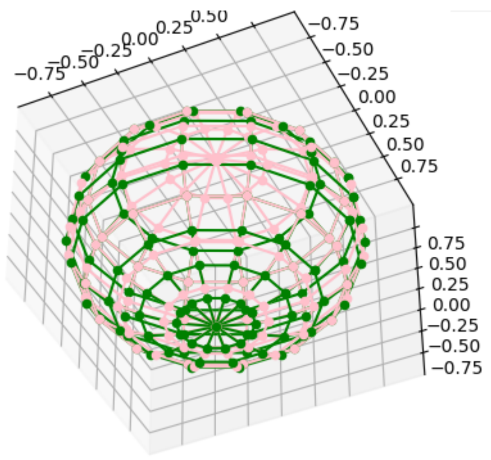


Figure 45: 3D candidate for a transformation which behaves like an adjective, preserving the norm but mapping the points on the spherical surface closer to each other. In this case the transformation is linear in the spherical coordinates but not in the Cartesian ones. The points tend to converge to the south pole of the sphere more and more as the parameters k tends to 1. Here we see the original points in pink and the image points in green.

It is important to notice that, even though this transformation is coordinate dependent and non linear, its action can locally be seen as a rotation whose angle and fixed axis depend on the position on the sphere. The matrix is

$$\begin{bmatrix} sin^2(\phi) + cos^2(\phi)cos(\gamma) & sin(\phi)cos(\phi)(cos(\gamma) - 1) & sin(\gamma)cos(\phi) \\ sin(\phi)cos(\phi)(cos(\gamma) - 1) & sin^2(\phi)cos(\gamma) + cos^2(\phi) & sin(\phi)sin(\gamma) \\ cos(\phi)sin(\gamma) & -sin(\phi)sin(\gamma) & cos(\gamma) \end{bmatrix} \tag{48}$$

where $\gamma(k, \theta) = k(\pi - \theta)$.

7.12.3 A model for N dimensions: The Procrustes Problem and The Kabsch's Algorithm

Once we posses a subset of active cluster centers and their images for a certain adjective, we might be interested in finding a linear transformation which approximates its non-linear action.
If we complete to a basis the subset of active cluster centers and we do the same for their images, in such a way that the additional vectors are left untouched by the transformation, we obtain to collections of n n-dimensional arrays, or in other words two data matrices A and B. We could then impose that each original vector is mapped into the correspondent image by defining the matrix

$$M = BA^{-1} \tag{49}$$

This procedure doesn't ensure the norm preservation: for example the transformation that maps the x-y-z orthogonal basis into the vectors that are columns of

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 1 \end{bmatrix} \tag{50}$$

changes the norm of a generic vector $\mathbf{x}$ into

$$\|\mathbf{y}\|^2 = \|\mathbf{x}\|^2 + xy + \sqrt{2}yz \tag{51}$$

.
We could therefore try with an orthogonal matrix, which automatically preserves the norms. The problem of finding the best orthogonal matrix which maps a given matrix A into another matrix B is called The Procrustes Problem and it is well known in the literature. If we limit our self to the special orthogonal group of rotations SO(N), we can use the previously mentioned Kabsch 's Algorithm [Kab76](section 7.9.2) to find the rotation matrix the most closely maps our original cluster centers into their images. This algorithm has recently become more popular especially in virology, where it has used in combination with other techniques to compare viral proteins such as the one of SARS-COV-2 and SARS [Com+].

The procedure gives us a global rotation which approximates the action of the adjectives on the noun vectors, and consists in the following algorithm

The algorithm:
Given two (N,M) data matrices A and B which have respectively the original the image vectors as rows it outputs a rotation matrix R (M,M) such that

$$R = \underset{\Omega}{argmin} \|\Omega A - B\|_F \tag{52}$$

, where $\|.\|_F$ denotes the Frobenius norm of matrices.
1) We center these matrices by subtracting their respective means

$$\begin{cases} P = A - \mu(\mathbf{A}) \\ Q = B - \mu(\mathbf{B}) \end{cases} \tag{53}$$

We define H as

$$H = P^{-1}Q \tag{54}$$

We apply the singular value decomposition to H in order to calculate U,S,V such that

$$H = USV^T \tag{55}$$

We check the chirality of the transformation by computing the quantity

$$d = sign(det(v^T u^T)) \tag{56}$$

and we prepare the correction diagonal matrix

$$D = diag(1, 1, .., d) \tag{57}$$

We finally output the rotation matrix

$$R = V^T D U^T \tag{58}$$

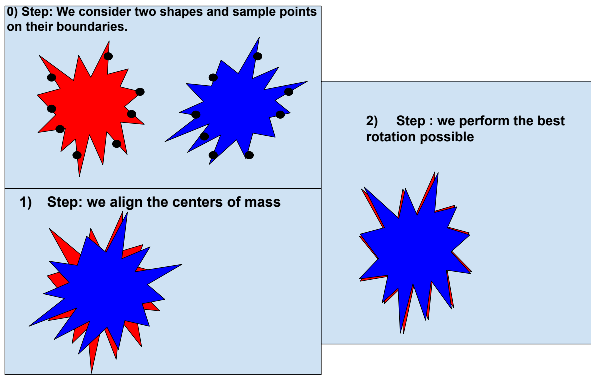

0) Step: We consider two shapes and sample points on their boundaries.
1) Step: we align the centers of mass
2) Step : we perform the best rotation possible

Figure 46: Illustration of the main steps of the Kabsch's algorithm.

7.12.4 Local Rotations

Since they are norm preserving, we can write each transformation as

$$f(\mathbf{x}) = e^{\sum_{i=0}^{N_G} \gamma(\mathbf{x})_i A_i} \mathbf{x} \tag{59}$$

where the A_i are again the $N_G = \frac{N(N-1)}{2}$ skew matrices which are the generators of the so_N Lie algebra of the rotation group. Differently from a normal rotation, which has the same expression in every point, the coefficients γ_i depend on the coordinates of the n-sphere.
In higher dimensions, we need to supply more real factors to fix a rotation than we did in the 3D situation we examined in section 7.13.2, thus simply establishing a rotation's axis and angle is insufficient. In n dimensions the number of real parameters that define a rotation is $\frac{N(N-1)}{2}$. The idea is to to approximate the adjective map with a set of rotations R_i, which are valid only for points belonging to a neighbourhood of each cluster center $\mathbf{X_{CMi}}$. If we add the constraint that $\mathbf{X_{CMi}}$ should be rotated into $\mathbf{Y_{CMi}}$, we only fix n-1 parameters. The remaining $\frac{(N-1)(N-2)}{2}$ real values gives us a lot of freedom on the orientation of the cluster on the surface. In 3D the only remaining degree of freedom is represented by a rotation angle around the fixed point $\mathbf{Y_{CMi}}$, but in 300 dimensions we can play with 44551 parameters. A possible procedure to additionally constraint the local rotation could be to impose the conservation of certain relations between the transformed clusters, at least approximately. We could, for example, try to preserve the orientation of the elements of each cluster with the respect to the center of all the others. For instance, if in the cluster containing words

like "man", "dude" and "father", the distance between the center of the cluster containing names of family members $\mathbf{X_{family}}$ and the embedding of "father" was the shorter than for the other two words, we would like to preserve this relation also in the transformed clusters. This means that we expect "old father" to be closer to $\mathbf{Y_{family}}$ than "old dude" or "old man".
For any given adjective, whose activity distribution has peaks on a subset of M cluster centers $\{\mathbf{X_{CMi}}\}$ with correspondent images $\{\mathbf{Y_{CMi}}\}$, we could output a set of M rotation matrices R_i which locally approximate the action of the adjective using the following algorithm
INPUT:
The list of vectors $\mathbf{X_{CM,i}}$ representing the centers of the clusters on the hyper-sphere and the set of all the word embeddings belonging to cluster i: $\mathbf{x_{ij}}$.
The list of vectors $\mathbf{Y_{CM,i}}$ representing the images of the the centers of the clusters under the action of a specific given adjective and the images of the points belonging to cluster i: $\mathbf{y_{ij}}$.

THE ALGORITHM:
In order to minimize the distance between the transformed cluster points and the images of their previously closest cluster centers, we adopted the following procedure, which exploits some concepts of the MDS and involves two rotations $R_{\alpha,i}$ and $R_{\beta,i}$for each cluster i for a given fixed adjective:

- 1) Given a cluster center $\mathbf{X_{CM,i}}$, we consider the list of the K closest cluster centers to it

$$C_i(k) = \{\mathbf{X_{CM,l_m(i)}}\}, m \in [1, K] \tag{60}$$

  For each word vector in the cluster i $\mathbf{x_{i,j}}$ we calculate its distances with the centers in C_k

$$d_{jl}(i) = \|\mathbf{x_{CM,l}} - \mathbf{x_{i,j}}\| \tag{61}$$

- 2) The training procedure described in the previous chapter explains how to find the image of each cluster center $\mathbf{y_{CM,i}}$. From the constraint $\mathbf{x_{CM,i}} \to R_{\alpha,i} \cdot \mathbf{x_{CM,i}} = \mathbf{y_{CM,i}}$ we can derive a random rotation $R_{\alpha,i}$ which satisfies it. In our case this constraint fixes only $300 - 1 = 299$ free parameters, so we are left with $\frac{N(N-1)}{2} - (N-1) = \frac{(N-1)(N-2)}{2} = 44551$. We can use this procedure for every cluster, applying to the i-th cluster center $\mathbf{x_{CM,i}}$ its first rotation $R_{\alpha,i}$, we obtain the transformed cluster centers $\mathbf{y_{CM,i}}$ and their correspondent points $\mathbf{y_{i,j}}$.

$$\mathbf{y_{i,j}} = R_{\alpha,i}\mathbf{x_{i,j}} \tag{62}$$

$$\begin{cases} \mathbf{y_{CM,i}} = R_{\alpha,i}\mathbf{x_{CM,i}} \\ \mathbf{y_{ij}} = R_{\alpha,i}\mathbf{x_{ij}} \end{cases} \tag{63}$$

  Our set of closest cluster centers will also be transformed into a set of image cluster centers

$$C_i'(k) = \{\mathbf{y_{CM,l_m(i)}}\}, m \in [1, K] \tag{64}$$

  where $\mathbf{y_{CM,l_m(i)}} = R_{\alpha,l_m(i)}\mathbf{x_{CM,l_m(i)}}$

- 3) For each transformed cluster i we now want to find the rotation $R_{\beta,i}$ around the new image center $\mathbf{y_{CM,i}}$ the minimizes the stress function

$$\begin{cases} Z_i[R(\gamma_1,..,\gamma_{N_G}),R_{\alpha,i}] = \sum_j \sum_l^K (d_{jl}(i)-d'_{jl}(i))^2 = \sum_j \sum_l^K (\|\mathbf{x_{CM,l}}-\mathbf{x_{i,j}}\| - \|R_{\alpha,l}\mathbf{x_{CM,il}} - R\cdot R_{\alpha,i}\mathbf{x_{ij}}\|)^2 \\ R_{\beta,i} = e^{\sum_{h=0}^{N_G}\gamma_h A_h} \\ (\gamma_1,...,\gamma_{N_G}) = \underset{z_1,..z_{N_G}}{argmin}(Z_i[R(z_1,..,z_{N_G}),R_{\alpha,i}]) \end{cases}$$

(65)

We define our locally approximating rotation for cluster i as

$$R_i = R_{\beta,i}R_{\alpha,i} \tag{66}$$

In order to completely fix the free parameters of these rotations, especially if we are working with small clusters, we can extend our constraint of distance preservation also to points that are outside of the considered cluster i but close enough to it.$\{\mathbf{x_{ij}}\} \longrightarrow \{\mathbf{x_{ij}}\} \cup \{\mathbf{x_{neigh(i)j}}\}$

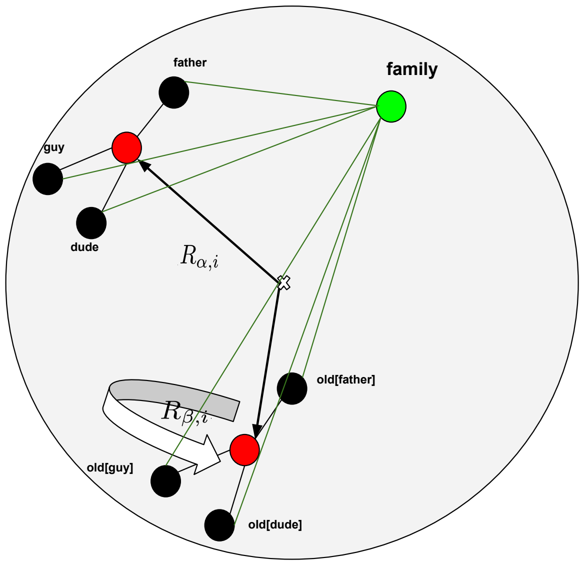


Figure 47: Scheme of the algorithm which performs local optimal rotations

7.12.5 Studying the case of a linear angle transformation

We can simplify our problem by studying the transformations of the adjectives at the level of angles. In this way, for each adjective we consider its subset of m active clusters and their images under its action. We then use the following formulas

$$\begin{cases} r = \sqrt{\sum_{i=1}^{n} x_i^2} \\ \phi_i = arccos\frac{x_i}{\sqrt{\sum_{j=i}^{n} x_i^2}}, for 0 < i < n-1 \\ \phi_{n-1} = \begin{cases} arccos\frac{x_{n-1}}{\sqrt{x_n^2+x_{n-1}^2}} if x_n \geq 0 \\ 2\pi - arccos\frac{x_{n-1}}{\sqrt{x_n^2+x_{n-1}^2}} if x_n \leq 0 \end{cases} \end{cases} \tag{67}$$

to go from the points on the n-sphere to the relative set of n-1 angles. The transformation can be non linear in the angles as well, but with this choice of coordinates the preservation of the norm is just r'=r=1. If we assume the transformation to be linear at the level of the angles, we can take a linearly independent subset of active cluster centers and of their correspondent images and complete them to a basis. Then the matrix of the liner transformation in the n-spherical coordinates is again given by

$$M = BA^{-1} \tag{68}$$

Once we know the entries of this matrix, we can use it to extract information about the transformation and also to indirectly calculate the image of every point under the action of the adjective: given a unit vector $\mathbf{x}$, we just need to find the angles associated to it $\theta(\mathbf{x})$, apply our matrix M to obtain the transformed angles $\alpha = M\theta$ and then finally calculate the correspondent unit vector $\mathbf{y}$.

$$\mathbf{y} = F(\mathbf{x}) \tag{69}$$

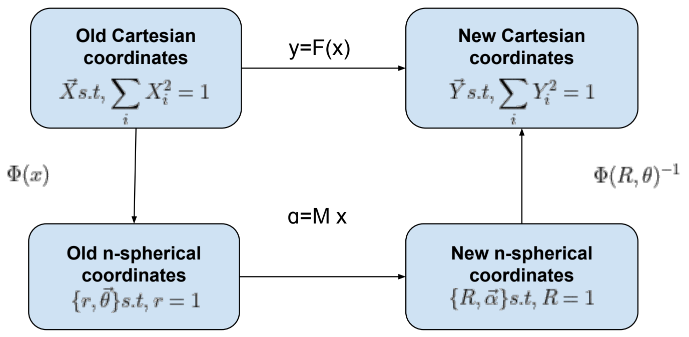

Old Cartesian coordinates
$\vec{X} s.t, \sum_i X_i^2 = 1$
y=F(x)
New Cartesian coordinates
$\vec{Y} s.t, \sum_i Y_i^2 = 1$
$\Phi(x)$
$\Phi(R,\theta)^{-1}$
α=M x
Old n-spherical coordinates
$\{r, \vec{\theta}\} s.t, r = 1$
New n-spherical coordinates
$\{R, \vec{\alpha}\} s.t, R = 1$

Figure 48: Scheme of the procedure involving a transformation which is linear at the level of angles

In N dimensions the new coordinates can be written as

$$\begin{cases} y_i = \prod_{j=1}^{i-1} sin((M\theta)_j)cos((M\theta)_i), 0 < i < n \\ y_n = \prod_{j=1}^{n} sin((M\theta)_j) \end{cases} \tag{70}$$

If the transformation $\alpha(\theta)$ is not linear, we can substitute the matrix M with the Jacobian $\frac{\partial \alpha_i}{\partial \theta_j}$. This approach automatically allows us to preserve the norm of the vectors, but has the disadvantage of not dealing directly with our word embeddings, which are written in Cartesian coordinates. A possible strategy to get around this obstacle is to exploit our knowledge of the transformation on a set of points on the n-sphere and the Jacobian Matrix. Since we posses a collection of active cluster centers and their relative images under the action of an adjective, we can approximate the image of any point word vector in the following way

- 1) We use an efficient algorithm for vector-distance minimization such as spatial.KDTree from the python library Scipy in order to find the closest cluster center to a given vector $\mathbf{x} \rightarrow \mathbf{X_{CMi}}$.
- 2) We calculate the displacement vector $\mathbf{d_i}(\mathbf{x})$ associated to $\mathbf{x}$
- 3) We consider the corresponding image cluster center $\mathbf{Y_{CM_i}}$.
- 4) We approximate the image word embedding with

$$\mathbf{y} = F(\mathbf{x}) = F(\mathbf{X_{CMi}} + \mathbf{d_i}(\mathbf{x})) \approx \mathbf{Y_{CM_i}} + J_F \cdot \mathbf{d_i}(\mathbf{x}) \tag{71}$$

where J_F is the Jacobian matrix of the vector field F associated to the adjective.

J_F is defined as

$$J_{Fij} = \frac{\partial y_i}{\partial x_j} \tag{72}$$

and it can be calculated using the chain rule as a product of three matrices

$$J_{Fij} = \frac{\partial y_i}{\partial x_j} = \sum_k \sum_l \frac{\partial y_i}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial \theta_l} \frac{\partial \theta_l}{\partial x_j} = J_\Phi M J_\Phi^{-1} \tag{73}$$

Where we used the fact that

$$J_{\Phi^{-1}} = J_\Phi^{-1} \tag{74}$$

The matrix J_Φ is the Jacobian matrix of the change of coordinates from the Cartesian to the n-spherical system. Its expression can be evaluated by induction with programs like Mathematica but it is also known in the literature as

$$\begin{bmatrix} c_1 & -rs_1 & 0 & ... & 0 & 0 \\ s_1c_2 & rc_1c_2 & -rs_1s_2 & ... & 0 & 0 \\ s_1s_2c_3 & rc_1s_2c_3 & rs_1c_2c_3 & ... & 0 & 0 \\ ... & ... & .. & ... & ... & ... \\ s_1...s_{n-2}c_{n-1} & rc_1s_2...s_{n-2}c_{n-1} & rs_1c_2s_3...s_{n-2}c_{n-1} & ... & rs_1s_2...c_{n-2}c_{n-1} & -rs_1s_2...s_{n-2}s_{n-1} \\ s_1...sn-1 & rc_1s_2...sn-1 & rs_1c_2s_3...s_{n-1} & ... & rs_1...c_{n-2}s_{n-1} & rs_1...s_{n-2}c_{n-1} \end{bmatrix} \tag{75}$$

where $c_i = cos(\theta_i)$ and $s_i = sin(\theta_i)$.
After the matrix product in equation (64) we set r=1.
The Jacobian Matrix is the best local linear approximation of the transformation induced by the adjective. The determinant of J_F is given by

$$det(J_F) = det(J_\Phi M J_\Phi^{-1}) = det(J_\Phi)det(M)det(J_\Phi^{-1}) = det(M) \tag{76}$$

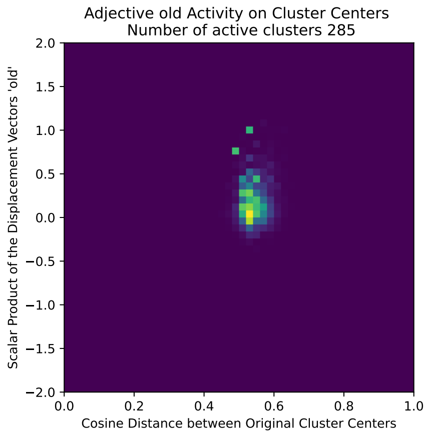
Where we used Binet Formula and the fact that $det(J_\Phi^- 1) = \frac{1}{det(J_\Phi)}$ if $det(J_\Phi) \neq 0$. This procedure is hence defined everywhere $J_\Phi \neq 0$.

7.12.6 2D Visualization

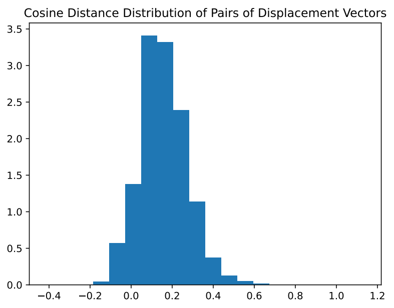
In the article [DX08] the authors describe an interesting way to visualize the action of a vector field with a 2D plot. The procedure consists in taking a set of pairs of random points from the original vector space, $(\mathbf{x_i}, \mathbf{x_j})$, and compute their correspondent images under the action of the vector field $(\mathbf{y_i}, \mathbf{y_j})$. In our case we define the value of our vector field at a given cluster center $\mathbf{X_{CMi}}$ as

$$\mathbf{V_i} = \mathbf{Y_{CMi}} - \mathbf{X_{CMi}} \tag{77}$$

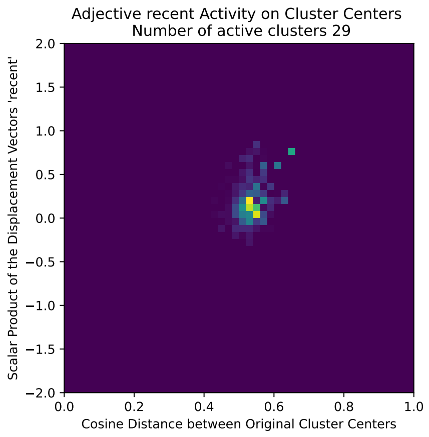
We then calculated the Euclidean distance between the points of each pair, the scalar product of the correspondent vectors and used these two values as 2D dimensional coordinates. Binning these points we obtained a 2D histogram that visually represents the action of the adjective to a certain extent. We applied this procedure to our data set and we produced the following 2D plot for the adjectives ”old” and ”new”. We slightly modified the procedure, since we used the cosine distance instead of the Euclidean distance, since they are proportional on a n-sphere, and we used the product of the probabilities (obtained from the normalization of the frequencies) associated to each cluster as weights in the histogram.



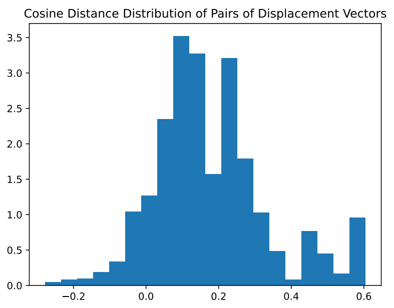
(a) Colormap of the 2D histogram associated to adjective ”old”



(b) Distribution of the scalar product of the displacement vectors associated to adjective ”old”



(c) Colormap of the 2D histogram associated to adjective ”recent”



(d) Distribution of the scalar product of the displacement vectors associated to adjective ”recent”

These plots can visually provide information about the mean and the variance of the distance distribution of the cluster centers and their images, but they are not efficient to compare different adjectives. We used the data set SimLex-999 [HRK14] containing 111 adj-adj comparisons based on human judgement as a target example and we confronted its values with our predictions. This data set can be represented by a symmetric matrix, which we display using a colormap as

More precisely, for each pair of adjectives we computed the chi squared distance

$$\chi^2_d(A,B) = \frac{1}{2}\sum_i \frac{(A[i]-B[i])^2}{(A[i]-B[i])} \tag{78}$$

between the flattened version of their normalized 2D histograms A and B. This number was considered as the prediction of our model for the similarity between the adjectives in the correspondent pair. We compared this values with the

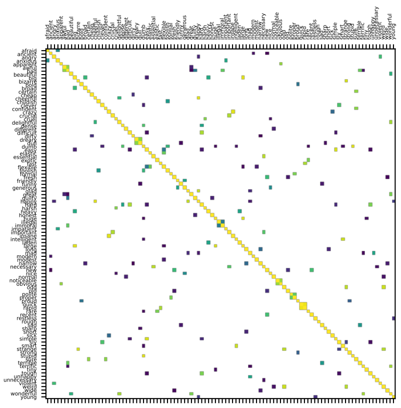


Figure 50: Color map of the matrix whose entries represent the values from the data set SimLex-999, based on human judgement.

ones of SimLex-999 data set using the R^2 test

$$\begin{cases} R^2(x,y) = 1 - \frac{SS_{res}}{SS_{tot}} \\ SS_{res} = \sum_i (x[i] - y[i])^2 \\ SS_{tot} = \sum_i (y[i] - \hat{y})^2 \\ \hat{y} = mean(y) \end{cases}$$

(79)

, for which we obtained a negative value of -0.91, which means that the prediction based on the χ^2 of the 2D histograms doesn't agree with the data set.

7.12.7 Mix of Von Mises-Fisher Distributions

As the number of dimension grows, an interesting alternative method to cluster points on an hyper-sphere and investigate the action of an adjective is introduced by the article "Clustering on the Unit Hypersphere using a mix of von Mises-Fisher Distributions". In this paper the authors describe a clustering procedure based on a probability distribution on the hypersphere, the von Mises-Fisher Distribution, defined as

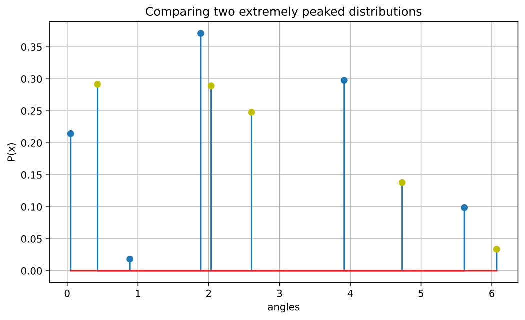
$$f(\mathbf{x}|\mu, \kappa) = c_d(\kappa) e^{\kappa \mu^T \mathbf{x}} \tag{80}$$

The quantity κ is the concentration parameter and characterizes how strongly the unit vectors drawn from the distribution converge to its mean μ. Given a fixed amount of distributions in the mix, it is possible to find the optimal set of

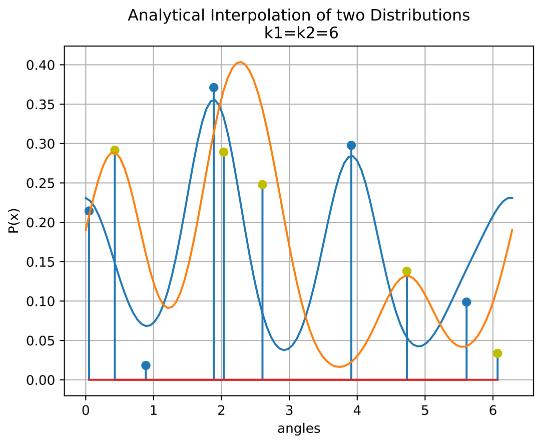
their parameters by maximizing the log-likelihood.
This mix of probability densities can in principle be used to describe the action of the adjective on the noun vectors in terms of changes in the mean vectors μs and the concentration parameters κs.
A possible approach would be to use to treat the activity distribution of an adjective as a probability distribution on the n-sphere. We could turn the pairs (frequency - cluster center) $(f_i, \mathbf{X_{CMi}})$ into the pairs (probability - cluster center) $(p_i, \mathbf{X_{CMi}})$ by normalizing the frequencies.
The next step would be to consider the correspondent image pairs $(p_i, \mathbf{Y_{CMi}})$, where $\mathbf{Y_{CMi}} = F_A(\mathbf{X_{CMi}})$, and the to treat them as respectively our weights and mean vectors for a mix of Mises-Fisher distributions:

$$f_A(\mathbf{x}|, \mu_\mathbf{s}, \kappa_s, p_s) = \sum_i p_i c_d(k_i) e^{\kappa_i \mu_\mathbf{i}{}^T \mathbf{x}} = \sum_i p_i c_d(k_i) e^{\kappa_i \mathbf{CMi}^T \mathbf{x}} \tag{81}$$

The reason why we believed this approach to be more appropriate is that the activity distribution of an adjective seen as just the set pairs $(p_i, \mathbf{Y_{CMi}})$ is a discrete probability distribution of vectors on the n-sphere. Comparing discrete distributions of points that live on a continuous manifold, which are equivalent to a weighted sum of delta distributions in the angles, would not give reasonable results. We are instead building a sort of analytical interpolation of this distributions, substituting the deltas with von Mises-Fisher distributions.
In 1D an intuitive explanation can be given by the following plots



(a)



(b)

Figure 51: Analytical Interpolation of the extremely peaked distribution using a mix of 1D von Mises-Fisher distributions with concentration parameters equal to 6.

As we can see from the plots, choosing the right value for the concentration parameters we can allow distributions whose peaks are not perfectly aligned but close enough to have a small point-wise distance. This procedure also makes the distance function less discontinuous, since, when dealing with delta functions of the angular values, even a slight relative rotation of the peaks could lead to a big change in the distance between two identical distributions of points on the n-sphere.
A possible interpretation for this probability densities could be the following: since each mean vector is the image of a cluster center and the relative weight is proportional to the activity frequency of the adjective on words belonging to the

associated cluster, if we choose the concentration parameters to be inversely proportional to mean radii of the clusters, we could consider the value of the density at a point $\mathbf{x}$ on the surface of the n-sphere as associated to the probability of the following event
”Given a uniformly random sampled noun embedding and a bi-gram adj+noun, the word embedding of the bi-gram belongs to an infinitesimal portion of n-sphere surface centered in $\mathbf{x}$.” Since each adjectives has non zero probabilities only on a specific subset of cluster centers, which it then rotates (not necessarily all by the same angles) on the n-sphere, we can think of an adjective as the sequential application of two maps on the uniform distribution over all the cluster centers given by

$$U(\mathbf{x}|, \mu_{\mathbf{s}}) = \frac{1}{n_{clusters}} \sum_i c_d(k_i) e^{\mathbf{X_{CMi}}^T \mathbf{x}} \tag{82}$$

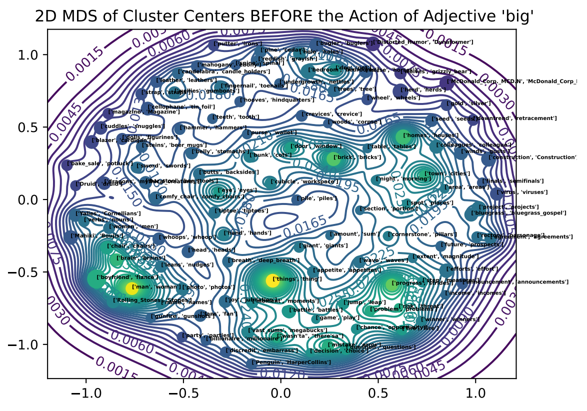
The 1D (we flattened the circumference) analogous of this distribution would be, using random points, the distribution in figure 21. The first transformation associated to the adjective modifies the probabilities p_i and the concentration parameters κ_s going from the uniform distribution U_A to

$$g_A(\mathbf{x}|, \mu_{\mathbf{s}}, \kappa_s, p_s) = \sum_i p_i c_d(k_i) e^{\kappa_i \mathbf{X_{CMi}}^T \mathbf{x}} \tag{83}$$

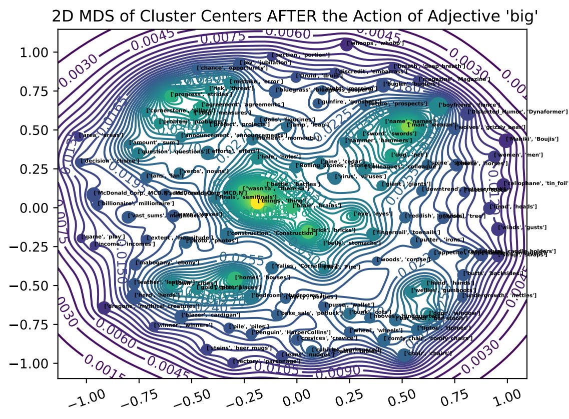
The 1D version of this transformation can be seen in subfigure 21(b). The second and last transformation rotates each peak to a different location on the n-sphere, which corresponds to the associated image center, without changing its height but possibly varying its κ.

$$f_A(g_A(\mathbf{x}|, \mu_{\mathbf{s}}, \kappa_s, p_s)) = \sum_i p_i c_d(k_i) e^{\kappa_i \mathbf{Y_{CMi}}^T \mathbf{x}} \tag{84}$$

Combining this new approach with the MDS technique described above, it is possible to represent the activity distribution of an adjective and the transformation it causes to the cluster centers as a 2D landscape. The color represents the height along the z axis, while the contours indicates sets of points with the same height. The sizes of each marker is proportional to the inverse of the radius of the correspondent cluster.

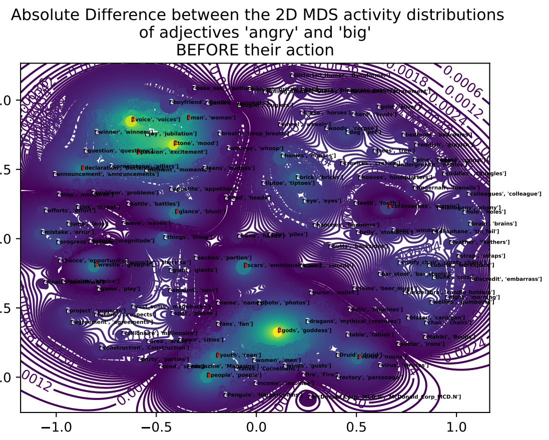


(a) Activity distribution of adjective ”big” before its action

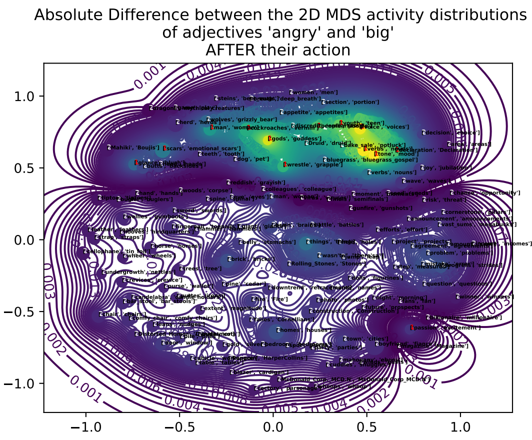


(b) Activity distribution of adjective ”big” after its action

It is also possible to use this 2D representation two compare the activity distributions of different adjectives, for example taking their average absolute difference. In the following two plots we show the result of such a comparison for the adjectives ”angry” and ”big”. The colormap again indicates the value on the z axis, normalized with the maximum value in the plot. The small colors numbers on the peaks 1 and 2 indicates respectively appartenance to distribution of adjective ”angry” and ”big”.



(a) Absolute difference between the 2D MDS of the activity distributions of adjectives ”big” and ”angry” before their action (on the original cluster centers)



(b) Absolute difference between the 2D MDS of the activity distributions of adjectives ”big” and ”angry” after their action (on the image cluster centers

8 Derivation Trees

8.1 Categorial Grammars

The problem of deriving all the possible interpretations of a given sentence has been widely and successfully studied by linguistics and logicians, such great historical figures as Kazimierz Ajdukiewicz, Yehoshua Bar-Hillel, Joachim Lambek and Richard Montague. The last two created their own formalism respectively called Lambek’s and Montague’s grammars as branches of categorial grammars. Categorial grammar is a family of formalisms in natural language syntax that share the central assumption that syntactic constituents combine as functions and arguments. In this work we propose a simplified version of the Lambek’s grammar, where words, according to the POS and TAG they are assigned by a machine learning algorithm, act like functions with an output and an arbitrary left and right input.

8.2 Ambiguities

As we saw in section 6, natural language shows several types of ambiguities at both syntactical and semantic level and it is therefore a complex problem to explore all the possible interpretations and represent only the most reasonable

ones.
In the next subsection we will show a possible attempt of solution to this problem which doesn't use any kind of quantum mechanical concept. We then will explore how we can exploit the idea of superposition of quantum states to speed up this classical algorithm, at least in principle.

8.3 A classical attempt at a solution to the derivation tree problem

Given a sentence which is grammatically correct in English, that we represent as an ordered list of words $[w_1, ..., w_n]$, we assign to each word a possible syntactical role according to its context. This can be done in multiple ways, in particular using recursive pre-trained neural networks which can assign a tag and a POS (part of speech) labels to each word in the input. In this work we will use the Python library SpaCy and more specifically the model ”en-core-web-sm”.
We start from the assumption that the words that carry most of the meaning of a sentence are verbs and nouns (personal pronouns are also included in this category), which we will consider as core logic types of our model. On the other hand, other grammar types like adjectives and adverbs have the role of modifying the meaning of respectively nouns and verbs, adding extra information on top of them. The remaining types like prepositions, particles and conjunctions are used to describe the relations between the core logic units in a clause and also between several clauses in a given complex sentence or period.
The natural consequence of this approach is that sentences are not longer represented by a single large vector but by a graph whose oriented edges represent the relations between the different components. Following again the reasoning of categorial grammars, we can think of each different syntactic type as a unit which can possibly accept a specific input from the left, one from the right and that always outputs a particular output. This concept can be visualized using an analogy with boxes and wires.
For example the word ”man” is unambiguously classified as a [tag=”NN”] type. This means that this word can be represented in our model as a box with only one output wire of kind ”N”.

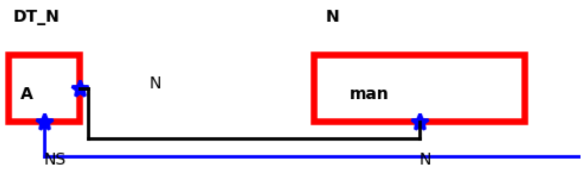


Figure 54: Example of a contraction that gives rise to a NS type. This picture, as well as all the other pictures of derivation trees, are generated by our program based on SpaCy and on the brute-force approach, which is described in this section.

In order to be connected to a verb and hence become the subject of a sentence, the word ”man” needs to change its output type from ”N” to ”NS”. Within the analogy of box connections, the box ”man” needs to connect its output wire to the input wire of the same kind belonging to another box. For example, if we consider the word ”a” we notice that it is also unambiguously classified as a [tag=”DT”] which means determiner. This type can correspond in our analogy to a box with one right input wire of kind ”N” and an output wire of kind ”NS”. This is coherent with the fact that ”a man” can be the subject of a sentence while ”man” alone can't.

If we now wish to complete this sentence we need a verb, for example "runs". The verb to run can be used both transitively and intransitively, therefore it can correspond in principle to two boxes:

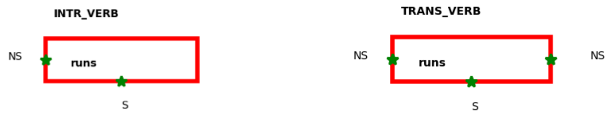


Figure 55: Example of box types associated to respectively transitive and intransitive verbs.

Which one of the two options is the right one can be often derived from the context, for example the presence of an additional "NS" type at the end of the sentence as "a marathon".

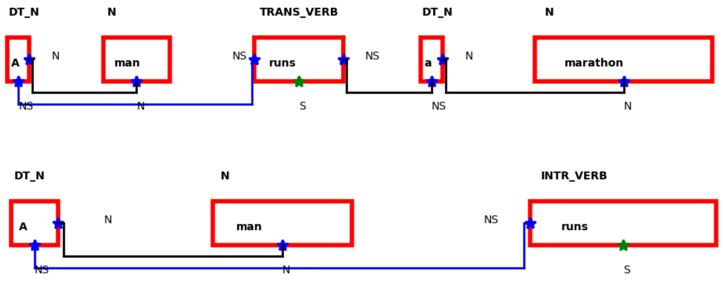


Figure 56: Example of different type assignments performed by the program. Each output wire can only connect with an input wire of the same type. This mimics the contractions between type.

When the context is not able to solve the degeneracy and multiple options are equally correct, we want our program to output all the possible valid interpretations.
Ambiguity can also emerge from the action of adjectives on nouns, as we saw in section 6. In this case, multiple options are possible at the semantic level and this degeneracy concerns the relation between words in the sentence and not the

interpretation of a single term.
For example, if we take the sentence ”rigorous mathematicians and physicists are numerous.”, two options are possible:

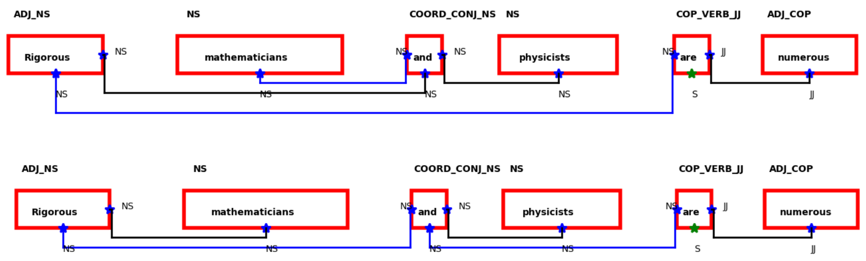


Figure 57: Different readings output by the program of the sentence ”rigorous mathematicians and physicists are numerous”. A probability score is assigned to each derivation by the program: respectively 0.807 and 0.193.8.3.1

An analogous kind of ambiguity comes from the action of adverbs on verbs, for example the sentence ” I drink and eat slowly.” has two syntactically correct interpretations:

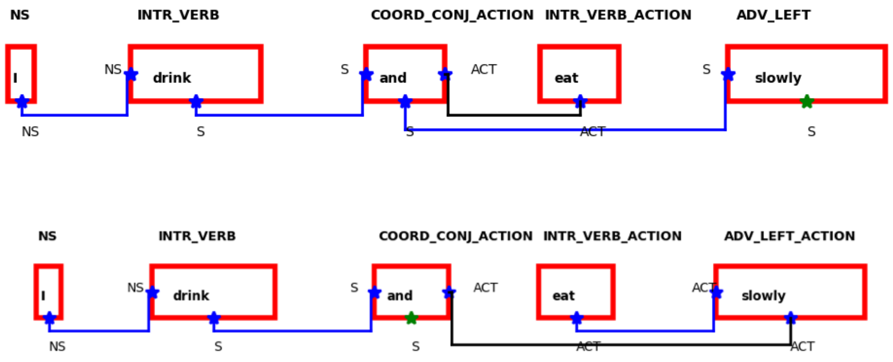


Figure 58: Example of a similar kind of ambiguity to the one we mentioned before. This time it involves the action of adverbs on verbs.

But we could also have a more subtle example with the sentence ”I quit smoking before lunch”, where ”before lunch” can refer in principle to both the actions of quitting and smoking.

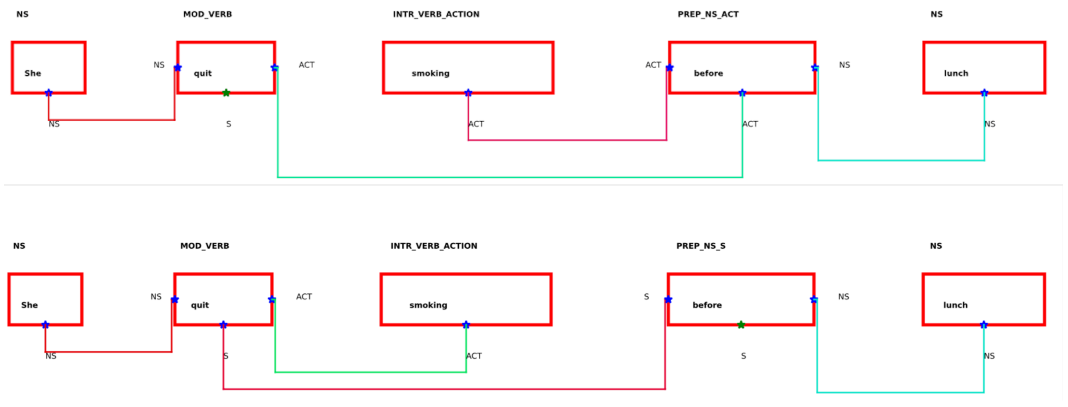


Figure 59: Another example of ambiguities related to the action of adverbs.

8.3.1 The algorithm

We will now introduce the algorithm that we used to obtain and display this derivation trees.
The first step of the algorithm applies the ”en-core-web-sm” model from SpaCy library to assign to each word in the sentence a pair TAG and POS.This model is quite fast and well optimized and has an accuracy of more than 90 percent(91.9 at parsing, 97.2 at tagging `https://spacy.io/usage/facts-figures`).
It is possible to assign to each combination a certain set of boxes of different types according to the following scheme:

TAG EXAMPLES	POS	SPECIFIC TEXT	BOX TYPES
NNS,NNP,NNPS,PRP I,you,she	any	else	NS
NNS,NNP,NNPS,PRP me,them	any	you,me,us,them	NS,ADV-L,ADV-L-ACT
VBG smoking,running	any	any	NS,IN-V-ACT,TR-V-ACT
NN smoking,running	any	else	N
NN yesterday,tomorrow,today	any	yesterday,tomorrow,today	ADV-R,ADV-L,ADV-R-ACT,ADV-L-ACT,NS
NN lunch,dinner,breakfast	any	lunch,dinner,breakfast	NN,NS
JJ,CD big,fat,three,two	any	any	ADJ-N,ADJ-NS,ADJ-COP
DT,PRP$ the	any	the	ADJ-NS,DT-N
DT,PRP$ a,an	any	a,an	DT-N
DT,PRP$ this,my,that	any	else	DT-N,ADJ-NS,PREP-NS-S
RB 's ,not	PART	any	ADV-L,ADV-R
RB slowly,lately	else	any	ADV-L,ADV-R,ADV-L-ACT,ADV-R-ACT
VBZ,VBP,VBD is,are	AUX	any	TR-V,IN-V,TR-V-ACT,IN-V-ACT,COP-V-NS,COP-V-JJ,COP-V-NS-ACT,COP-V-JJ-ACT
VBZ,VBP,VBD seem,resemble,appear	else	seem,resemble,appear etc	TR-V,IN-V,TR-V-ACT,IN-V-ACT,COP-V-NS,COP-V-JJ,COP-V-NS-ACT,COP-V-JJ-ACT
VBZ,VBP,VBD love,hate,prefer	else	love,hate,prefer etc	TR-V,IN-V,TR-V-ACT,IN-V-ACT,MOD-V
VBZ,VBP,VBD love,hate,prefer	else	bring,show,give etc	TR-V,IN-V,TR-V-ACT,IN-V-ACT,DOUB-TRANS
VB see,go(base form)	else	any	TR-V-ACT,IN-V-ACT
VB be(base form)	AUX	any	COP-V-NS-ACT,COP-V-JJ-ACT
VB seem,resemble(base form)	else	seem,resemble,appear	COP-V-NS-ACT,COP-V-JJ-ACT
MD can,might,may	any	any	IN-V,MOD-V
WDT that	any	any	ADJ-CL-CONJ
WP what	any	any	NOUN-CL-CONJ
IN that	ADP	any	PREP-NS-ACT,PREP-NS-S
IN that	else	that	NOUN-CL-CONJ
IN that	else	else	ADV-CL-CONJ
TO I want to go	any	any	PREP-TO
JJR you are faster than me	any	any	JJR,ADJ-COP,ADVR,ADV-L,ADV-L-ACT

In this way a sentence or list of words is converted to a list of sets, each containing one or more boxes. For example, a word like ”man”, which is a countable noun, can only be assigned a set containing one box of type ”N”, whereas a word like ”eating” (if classified as tag VBG) would correspond to the set of boxes [”NS”,”IN-V-ACT”,”TR-V-ACT”] because it can be interpreted either as a noun (”eating is indispensable”) or as both a intransitive or transitive modifiable action (”I was eating more slowly than usual ” or ”I saw that he was eating chips” . Since it is in principle non trivial to determine which connections between different boxes are allowed by the grammar, we proceed with a brute-force approach which will be then refined and filtered using selection rules.
From the list S of sets representing all the possible grammatical roles of the words in the sentence, which is therefore made of N elements where N is the length of the sentence, we produce a list of lists I_i, each of length N, that represent the possible interpretations. In each of those lists the type of every box is fixed. The amount of such interpretations we produce is determined by the product of the cardinalities of all the sets in the original sentence list. From now on we will call this sets of possible types TextUnits.

$$|\{I_i\}| = \prod_{l=0}^{N} |S[i]| \tag{85}$$

.
For example: "I always eat an apple" when analyzed by spaCY outputs the following pos-tags:

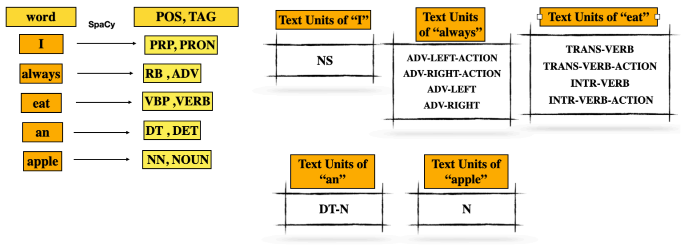


Figure 60: Example of POS , TAG and text units assignment

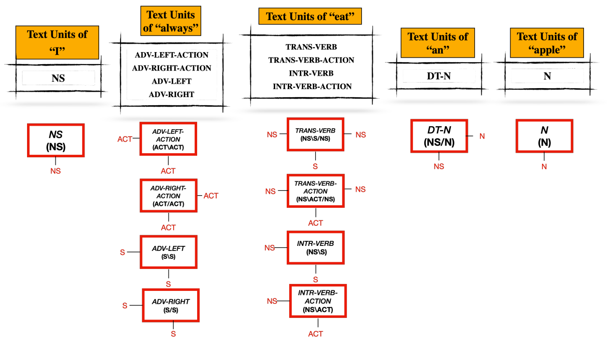


Figure 61: Example of box types associated to each text unit

Each of the boxes in these TextUnits has different kinds of wires: they can be left or right input wires (which are optional) or output wires (mandatory). Each wire has also a type that belongs to the set of basic types

$$B.T. = \{ACT, N, NS, S, JJ, COMP\} \tag{86}$$

Following our example we have the possible $1 \cdot 4 \cdot 4 \cdot 1 \cdot 1 = 16$ interpretations:
1)$(\ (|||NS + |||), (ACT - |||ACT + |||), (NS - |||S + |||NS-), (|||NS + |||N-), (|||N + |||))$
2)$((|||NS + |||), (ACT - |||ACT + |||), (|||ACT + |||NS-), (|||NS + |||N-), (|||N + |||))$
3)$((|||NS + |||), (ACT - |||ACT + |||), (NS - |||S + |||), (|||NS + |||N-), (|||N + |||))$
4)$((|||NS + |||), (ACT - |||ACT + |||), (|||ACT + |||), (|||NS + |||N-), (|||N + |||))$
5)$((|||NS + |||), (|||ACT + |||ACT-), (NS - |||S + |||NS-), (|||NS + |||N-), (|||N + |||))$
6)$((|||NS + |||), (|||ACT + |||ACT-), (|||ACT + |||NS-), (|||NS + |||N-), (|||N + |||))$
7)$((|||NS + |||), (|||ACT + |||ACT-), (NS - |||S + |||), (|||NS + |||N-), (|||N + |||))$
8)$((|||NS + |||), (|||ACT + |||ACT-), (|||ACT + |||), (|||NS + |||N-), (|||N + |||))$
9)$((|||NS + |||), (S - |||S + |||), (NS - |||S + |||NS-), (|||NS + |||N-), (|||N + |||))$
10)$((|||NS + |||), (S - |||S + |||), (|||ACT + |||NS-), (|||NS + |||N-), (|||N + |||))$

$11)((\|\|\|NS+\|\|\|),(S-\|\|\|S+\|\|\|),(NS-\|\|\|S+\|\|\|),(\|\|\|NS+\|\|\|N-),(\|\|\|N+\|\|\|))$
$12)((\|\|\|NS+\|\|\|),(S-\|\|\|S+\|\|\|),(\|\|\|ACT+\|\|\|),(\|\|\|NS+\|\|\|N-),(\|\|\|N+\|\|\|))$
$13)((\|\|\|NS+\|\|\|),(\|\|\|S+\|\|\|S-),(NS-\|\|\|S+\|\|\|NS-),(\|\|\|NS+\|\|\|N-),(\|\|\|N+\|\|\|))$
$14)((\|\|\|NS+\|\|\|),(\|\|\|S+\|\|\|S-),(\|\|\|ACT+\|\|\|NS-),(\|\|\|NS+\|\|\|N-),(\|\|\|N+\|\|\|))$
$15)((\|\|\|NS+\|\|\|),(\|\|\|S+\|\|\|S-),(NS-\|\|\|S+\|\|\|),(\|\|\|NS+\|\|\|N-),(\|\|\|N+\|\|\|))$
$16)((\|\|\|NS+\|\|\|),(\|\|\|S+\|\|\|S-),(\|\|\|ACT+\|\|\|),(\|\|\|NS+\|\|\|N-),(\|\|\|N+\|\|\|))$

We then filter out all the terms where the different types of wire don't cancel each other out, with the exception of the additional final output S of the sentence.
We are left with only one result:
$((\|\|\|NS+\|\|\|),(\|\|\|S+\|\|\|S-),(NS-\|\|\|S+\|\|\|NS-),(\|\|\|NS+\|\|\|N-),(\|\|\|N+\|\|\|))$

which corresponds to the only possible derivation tree

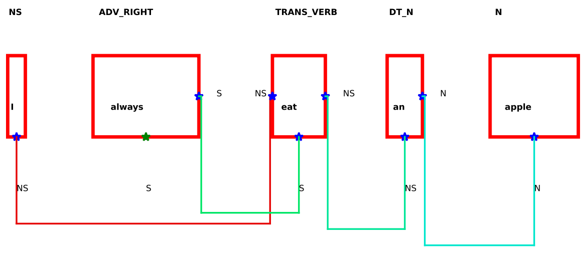


Figure 62: Visual representation of the only derivation tree that is possible for the sentence " I always eat an apple."

Once we have all the coherent interpretations we need to perform all the possible contractions between their boxes, which means trying all the possible connections between pairs of wires of the same type that at the end leave out only an output wire of type S (the sentence wire). Since some of these contractions are non associative, we treat all of them as non associative. This problem becomes analogous to the one of contracting matrices with different dimensions. In that case [SW19], the matrix chain multiplication algorithm finds the optimal parenthesization of the matrices such that the number of scalar multiplications is minimized (and hence the computational cost). In our case the parameter we are interested in is the total probability of a given reading of a sentence, which is defined as the product of all the probabilities related to the single contractions in that specific derivation. More precisely, as we will better see later in the quantum approach, it is possible to associate to any given contraction of wires a certain positive number which is smaller than one and that is proportional to the similarity (for example cosine similarity in the classical case) between the representations of the two words involved in the contraction. The product of this numbers is associated to the derivation which contains the relative contractions. If we then normalize all these values we can interpret them as the probability for each reading to be correct one.
In the case of boxes and wires, for a given interpretation containing N boxes of assigned type, the amount of possible contractions is given by the Catalan number [Sta15] C_{N-1}, where

$$C_n = \frac{1}{n+1}\binom{2n}{n} \tag{87}$$

For example, if we have three words associated to matrices A_1, A_2, A_3 the different contractions are:
$((A_1, A_2), A_3)$ and $(A_1, (A_2, A_3))$
In case of four matrices we would have 5 possible contractions :
$(((A_1, A_2), A_3), A_4)$, $((A_1, (A_2, A_3)), A_4)$, $(A_1, (A_2, (A_3, A_4)))$, $((A_1, A_2), (A_3, A_4))$, $(A_1, ((A_2, A_3), A_4))$ and so on .
In order to perform the contractions and verify the validity of each step in the derivation tree we introduce a new object called SemUnit. A SemUnit contains a certain number of boxes and information about the existing connections between them. At the beginning of the calculation, starting with a given coherent interpretation of length N, we define N different SemUnit, each containing one box. As the contractions are performed the SemUnits merge, giving rise to larger SemUnits with less wires available. Every time we perform a contraction we attach one output wire from one box to the input wire of another box, accordingly to the left right order inside the sentence. In other words, boxes can connect their right input wires only to wires which belong to boxes on their right and the same stands for the left input wires. Whenever two boxes belonging to different SemUnits are connected, the corresponding SemUnits merge until either there is only one left, containing N boxes, or a contradiction occurs and the derivation is classified as wrong. The algorithm that performs the contractions is recursive, but it can be improved storing partial results in an array so that its final cost is $O(N^2)$, analogously to what is done for the matrix chain multiplication [SW19].
Furthermore, as we mentioned before, if we associate to every box a vector representation such as a word embedding, every time we perform a contraction we can keep track of the likelihood of that connection by taking the cosine similarity between the two vectors. We then associate to the new SemUnit coming out of the contraction the average between the two vectors corresponding to the contracted words. In this way, contraction after contraction in the derivation tree, this real numbers are multiplied until they provide a certain final value for the derivation. If we then normalize these values through all the possible derivation trees (summing up the ones that are equivalent to each other) we get an approximation of the probability for each contraction to be the right one .

8.3.2 Beyond the brute-force approach

The above mentioned algorithm exploits some knowledge about the grammar types of the words but still blindly tries to perform all the possible contractions between text units.
A wiser method would be to detect the particular combination of word types that generate ambiguities in a sentence, at least the ones related to different contractions between adjectives and nouns or between adverbs and verbs. We could then consider this binary choices as the only source of ambiguities in the derivation apart from the ones at the word level. If we examine the example statement ”old woman and men retire,” we can see how two interpretations are feasible since in one case only old women are permitted to retire, whilst in the other both women and men must be relatively elderly. If we think about it, the second option seems more likely according to western social customs, so we expect a model which has been trained on a large corpus to reflect this preference in terms of probabilities.
This ambiguity in the derivation trees is conceptually identical to the one in ”rigorous mathematicians and physicists are numerous” since in both cases we have and adjective followed by two nouns in an AND.
At the level of box wires the sentence ”old women and men retire” reads from left to right

$$O_0R_0 - O_0 - L_0O_0R_0 - O_0 - L_0O_1 \tag{88}$$

where R indicates a right input wire, O an output wire and L a left input wire and the subscript labels the different types of wires. We notice that each ambiguity comes from the presence of the string ROL because the output type could in principle contract to the left or to the right. Every time we make a choice in that sense we explore the branch of a binary tree, which has as many levels as the amount of ROL strings which appear through the whole derivation. For

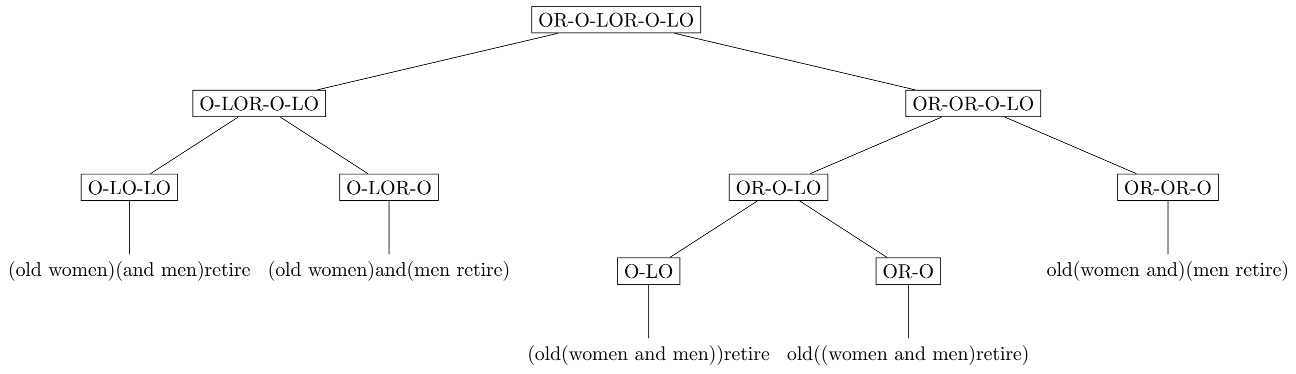

OR-O-LOR-O-LO
O-LOR-O-LO
OR-OR-O-LO
O-LO-LO
O-LOR-O
OR-O-LO
OR-OR-O
(old women)(and men)retire
(old women)and(men retire)
O-LO
OR-O
old(women and)(men retire)
(old(women and men))retire
old((women and men)retire)

Figure 63: Example of the derivation tree of a sentence.

example in our case we have:

The presence of two ROL strings in this assignment of box types to the words in the sentence leads to four possible interpretations of its meaning, which then become 5 because another ROL string is generated during the derivation. If we also consider the label of each wire, we can immediately discard interpretations 2,4 and 5 since they connect an S output type (”men retire”) to a NS right input type (”and”). We can see how this approach explores a lower amount of possibilities than the previous brute-force method, which would require $C_{5-1} = C_4 = 14$ possibilities.

8.3.3 Compositional sistributional semantics using tensor contraction

As explained in the PhD thesis ”Quantum Distributional Semantics” by doctor A.D. Correia [ADC22] as well as in the article [CMS22] , it is possible to derive the representation of the meaning of a sentence from the initial representation of its components through index contraction. More precisely, we can associate to every word a tensor representation, so that a given sentence would correspond to a their tensor product. Each of these tensors has a different amount of indices which depends on the word type. In analogy with the approach we explained in the previous section, this indices correspond to the wires of the boxes. For example, a NS type could be represented by a vector, while and adjective, which has type NS/NS, is associated to a matrix. Once we have this representation for the whole sentence we can interpret all the different possible contractions of pairs of indices as different readings of the former. Moreover, we could take track of the likelihood of each of this contraction with respect to a specific model (and hence to a specific corpus the model was trained on) in order to assign a probability for each reading to be the correct one given the context. For instance, in the sentence ”I go to school every morning with a small bag.” the complement ”with a small bag” could in principle refer to the action of me going to school or to the noun ”morning”. The first interpretation is obviously far more reasonable, since mornings don’t carry bags, so much that we don’t even notice the presence of the other alternative, but this is not necessarily so obvious for a machine. A more careful ordering of the sentence would probably fix the ambiguity, such as ”I go to school with a small bag every morning”, but for a human reader this is clearly not necessary in order to grasp the correct interpretation. When the complexity of the sentence grows, or when we start to chain phrases together to build a period, the amount of alternative interpretations can become relevant.

Going back to our prototype sentence ”Old women and men retire”, the four interpretations we previously derived would correspond to the following tensor contractions respectively:
(Old women)(and men)retire$\longrightarrow O_{ij}W_jA_{ikl}M_lR_{kp}$
(Old women)and(men retire)$\longrightarrow O_{ij}W_jA_{ipl}M_kR_kl$
(Old(women and men))retire$\longrightarrow O_{ij}W_kA_{kjl}M_lR_ip$
Old((women and men)retire)$\longrightarrow O_{ij}W_kA_{kpl}M_lR_pj$
Old(women and)(men retire)$\longrightarrow O_{ij}W_kA_{kjl}M_pR_pl$
In the following section we will show a way to implement this derivations in a quantum setting.

8.4 A quantum version of this algorithm

The procedure previously described seems to give reasonable results for the parsing of ordinary sentences and could be improved in order to cover less trivial construct of the English language. This could be achieved by, for example, extending the set of basic types or introducing a neural network that classifies transitive and intransitive verbs upstream. However, certain features of quantum computation, such as the parallelism due to quantum superposition of states, could possibly provide a quadratic speed up for the execution time and an exponential advantage in terms of memory, if certain conditions are fulfilled.

8.4.1 Encode classical information into quantum states

In the article [Sah22], the authors describe an advanced way to encode arrays of real numbers, for example our word vectors, into the amplitude of the quantum state of a quantum register($\mathcal{N}$ is normalization factor):

$$\{c_0, c_1, ..., c_{N-1}\} \longrightarrow |\Psi\rangle = \frac{1}{\mathcal{N}} \sum_{k=0}^{N-1} c_k|k\rangle \tag{89}$$

On the one hand, such an encoding provides an exponential saving of memory, but on the other hand it is known to require an exponential amount of single qubit and two qubit gates to act on an arbitrary input array. In this paper Sahel Ashhab et al introduce a new method which, assuming the non sparsity of the array values, shows a polynomial resource scaling. The non sparsity (or density) of the input array is defined as

$$\rho = \frac{1}{2^n} \sum_{k=0}^{N-1} |\frac{c_k}{c_{max}}|^2$$

(90)

This parameter takes its maximum value $\rho = 1$ when all $|c_k|$ are equal, while $\rho \ll 1$ for sparse data, where only a few c_k values are of the same order as c_{max}.
Their procedure is based on partial C-NOT gates and probabilistic projections onto the desired state and proves to be quite promising for quantum machine learning. The probability of success of this optimized algorithm and the time

needed to prepare the state are related to the maximum acceptable relative encoding error $\epsilon = max_k\{\epsilon_k\}$ and the dimension of the input $N = 2^n$:

$$\begin{cases} P_{success} \propto \rho\epsilon \\ T \propto \frac{\log n}{\rho\epsilon} \end{cases} \tag{91}$$

Given the intensive research and important results achieved in this field, we will now assume the existence of an encoder gate which takes the classical real representation of our tensors as input and outputs their correspondent quantum encoding state.
For example, for vectors we have

$$\mathbf{v} = (v_0, v_1, ..., v_{N-1}) \longrightarrow |v\rangle = \frac{1}{\mathcal{N}_v}\sum_{k=0}^{N-1} v_k|k\rangle \tag{92}$$

whereas for a matrix :

$$A \longrightarrow |A\rangle = \frac{1}{\mathcal{N}}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} A_{ij}|i\rangle|j\rangle \tag{93}$$

8.4.2 A quantum circuit to contract indices

The paper ”Tensor Contraction for Language Processing with Quantum Computers” by L. Huygens, A. D. Correia, and H. T. C. Stoof introduces two different alternatives for a quantum circuit that performs the equivalent of an index contraction at a state level. The first circuit, which is also the one used by doctor Correia in her Phd thesis, relies on the measurement of the permutation operator P :

$$P_{kl}\,|i_1\rangle \,..\, |i_k\rangle \,..\, |i_l\rangle \,...\, |i_n\rangle = |i_1\rangle \,..\, |i_l\rangle \,..\, |i_k\rangle \,...\, |i_n\rangle \tag{94}$$

on the desired pair of quantum wires, which are in turn associated to the tensor indices we want to contract. If we encode the tensor representing the components of our sentence, for example an adjective ”Red” and a noun ”apples”, using a certain basis

$$\begin{cases} |Red\rangle = \sum_i \sum_j r_{ij}\,|i\rangle\,|j\rangle \\ |Apples\rangle = \sum_i a_i\,|i\rangle \end{cases} \tag{95}$$

After a measurement of the first qubit from the left collapsed it into the state $|i\rangle$, the expected value of the P_{23} operator is :

$$\begin{aligned} [\sum_{j'k'} r^*_{ij'}a^*_{k'}\,\langle i'|\,\langle j'|\,\langle k'|]P_{23}[\sum_{jk} r_{ij}a_k\,|i\rangle\,|j\rangle\,|k\rangle] = \\ \sum_{j'k'jk} r^*_{ij'}a^*_{k'}r_{ij}a_k\,\langle ij'k'|ikj\rangle = \\ (\sum_k r^*_{ik}a_k)(\sum_j r_{ij}a^*_j) = \;|\sum_k r^*_{ik}a_k|^2 \end{aligned}$$

(96)

If we restrict ourselves to work with real values, the term $\sum_k r^*_{ik} a_k = \sum_k r^{ik^{a_k}}$ corresponds to the contraction between the tensor associated to "red" and the one associated to "apples". This expectation value can be then estimated by performing several shots of the Hadamard test circuit [AJL05]:

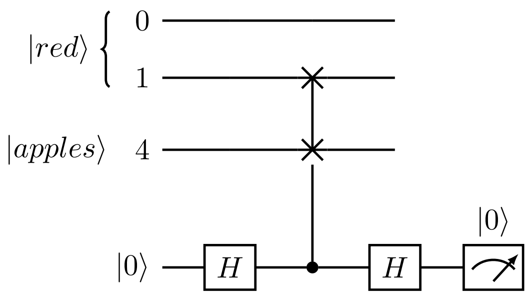


Figure 64: Example of the Hadamard test. We ran this circuit multiple times to extimate $Re \langle\phi| U |\phi\rangle$, in this case the U is the Permutation operator between two indices or Swap.

of our measure the expected value of the permutation operator, applied to the quantum wires to be contracted we get. A clear example, taken from pages 156-158 of the thesis, is given by figure 65, where the noun phrase rigorous mathematicians and physicists is given as input to the two circuits:

These two circuits correspond the only two grammatically correct derivations of the sentence (with wire string OR-O-LOR-O) and they give as output respectively the states:

1)O-LOR-O $\longrightarrow$(rigorous mathematicians)and(physicists) $\longrightarrow \sum\limits_{i,j,l,n} r_{ij} m_l a^*_{ljn} p^*_n |n_i\rangle$
and
2)OR-OR-O $\longrightarrow$rigorous (mathematicians and) physicists $\longrightarrow \sum\limits_{j,l,m,n} r^*_{lj} m_j a_{lmn} p^*_n |n_m\rangle$

Since the quantum formalism works with complex vectors, we can notice how complex conjugation plays a role in this method. However, in this work we will restrict our considerations to real valued tensors.
Furthermore, in a more recent paper than incorporated inside doctor Correia's work, the authors introduce a technique to compute both the results on the same circuit, using an additional "ancilla" control qubit and two C-SWAP gates.

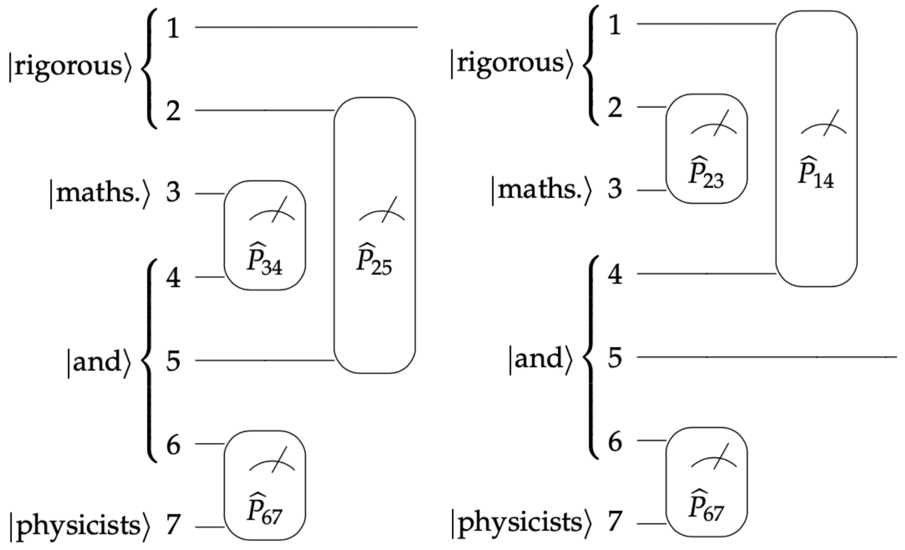


Figure 65: Example of the application of the permutation operator to perform different readings of the noun phrase ”rigorous mathematicians and physicists”.

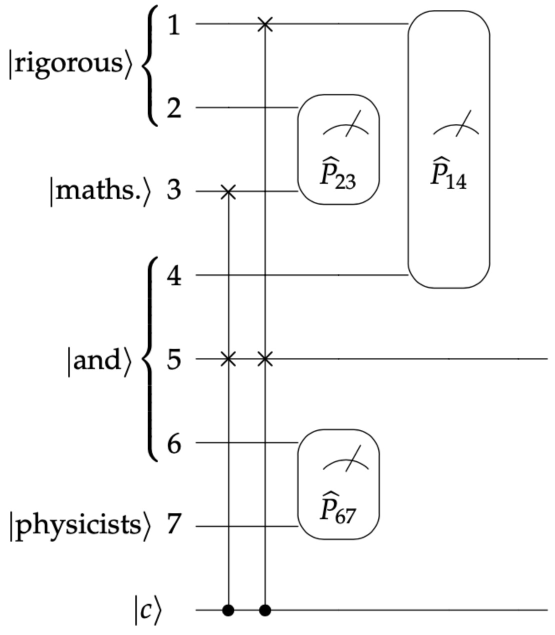


Figure 66: Circuit that exploits the C-SWAP gate to perform multiple readings of the sentence.

If we prepare the ancilla qubit in a superposition of the computational basis states ($|\alpha|^2 + |\beta|^2 = 1$)

$$|c\rangle = \alpha|0\rangle + \beta|1\rangle \tag{97}$$

we can associate the two orthonormal basis vectors to the two final states 1) and 2).
If we then measure it, our final output state, carried by quantum wire 5, collapses to the desired alternative with a probability that is equal to respectively $|\alpha|^2$ and $|\beta|^2$.

8.4.3 An alternative to the permutation operator

The second circuit is based on the Inverse Bell transformation operator and represents an interesting alternative since it only involves single qubit measurements in the computational basis. The basic version of the circuit contracts only a pair of indices:

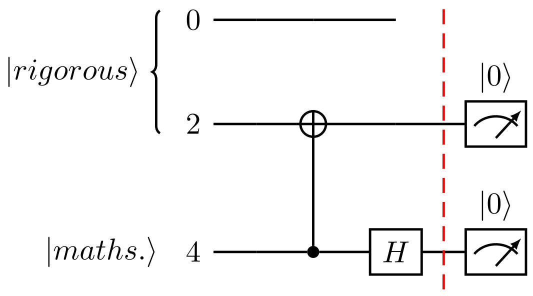


Figure 67: Contractor Circuit based on the inverse Bell transformation.

If we start with encodings

$$\begin{cases} |rigorous\rangle = \sum_i \sum_j R_{ij}|i\rangle|j\rangle \\ |maths\rangle = \sum_l M_l|l\rangle \end{cases} \tag{98}$$

At the dashed line the state of the system can be written as

$$|\Psi\rangle = |\Psi_{00}\rangle + |\Phi\rangle$$

(99)

where

$$|\Psi_{00}\rangle = \frac{1}{\sqrt{2}} \sum_k [(\sum_i R_{ki} M_i)|k00\rangle]$$

(100)

We can see how the amplitude of the state where both qubits 1 and 2 are in $|0\rangle$ stores the contracted tensor $(\sum_i R_{il} M_l)$. Since this circuit was going to be our basic step in the following calculation, we investigated its behavior through a simulation on Qiskit.
We choose the matrix R to be a rotation of a variable angle θ and M to be the zero basis vector:

$$\begin{cases} |rigorous\rangle = \frac{1}{\sqrt{(2)}}[cos(\theta)|00\rangle - sin(\theta)|01\rangle + sin(\theta)|10\rangle + cos(\theta)|11\rangle] \\ |maths\rangle = |0\rangle \end{cases} \tag{101}$$

so that the expected state should be

$$\begin{cases} |\Psi_{00}\rangle = \frac{1}{\sqrt{2}} \sum_k [(\sum_i R_{ki} M_i)|k00\rangle] = \frac{1}{2}[cos(\theta)|0\rangle + sin(\theta)|1\rangle]|00\rangle \end{cases} \tag{102}$$

which has a probability of being measured $P_{00} = \frac{1}{4}$

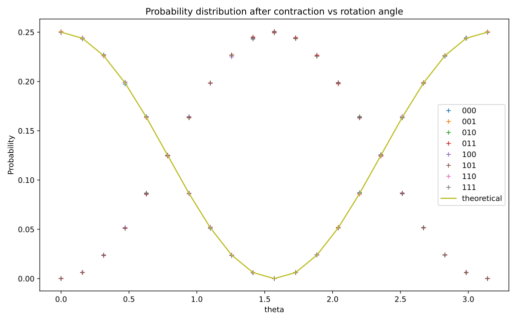


Figure 68: Plot of the estimated probability distribution of the desired state. The simulation using Qiskit was run each time with 100k shots and averaged over 5 attempts.

8.4.4 Generalization to N dimensions

In order to generalize to N dimensions we first have to notice that, given a n-qubit register, we can store real vectors up to dimension 2^n. For example, if we want to work in 3 dimensions, we can use two qubits, store the array entries in the first three components and set the fourth one to zero. A three dimensional generalization of the circuit above would be given by the following

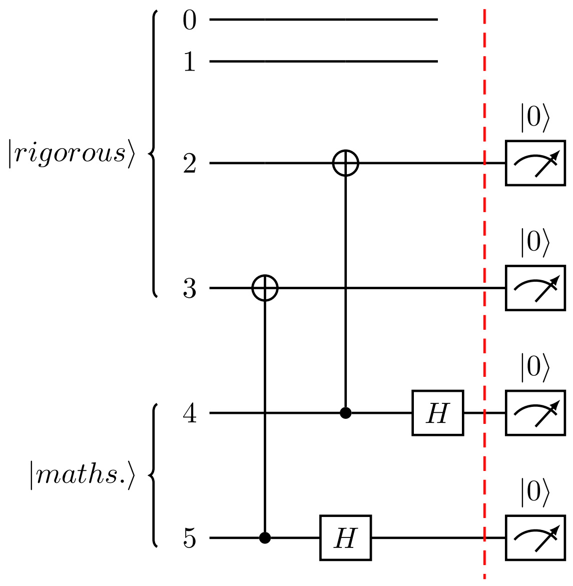


Figure 69: Contractor circuit in N=4 dimensions.

With the initial states given by (in the computational basis)

$$\begin{cases} |rigorous\rangle = \frac{1}{2}[sin(\theta)cos(\phi)|0000\rangle - sin(\phi)|0001\rangle - cos(\theta)cos(\phi)|0010\rangle + 0|0011\rangle + \\ \qquad sin(\theta)sin(\phi)|0100\rangle + cos(\phi)|0101\rangle - cos(\theta)sin(\phi)|0110\rangle + 0|0111\rangle + \\ \qquad cos(\theta)|1000\rangle + 0|1001\rangle + sin(\theta)|1010\rangle + 0|1011\rangle + \\ \qquad 0|1100\rangle + 0|1101\rangle + 0|1110\rangle + 1|1111\rangle] \\ |maths\rangle = |10\rangle \end{cases} \tag{103}$$

so that the expected state at the dashed line should be

$$\begin{cases} |\Psi_{00}\rangle = \frac{1}{2}\sum_{kl}[(\sum_{ab} R_{klab}M_{ab})|k\rangle|l\rangle|0000\rangle] = \frac{1}{2}[sin(\theta)cos(\phi)\,|00\rangle + sin(\theta)sin(\phi)\,|01\rangle + cos(\theta)\,|10\rangle]\,|0000\rangle \end{cases} \tag{104}$$

However, when simulating with Qiskit we need to be careful about the peculiar order in which the states are initialized(the first digit is the least relevant) :

If we wish to encode the desired rotation matrix

$$R = \begin{bmatrix} \cos(\theta)cos(\phi) & -sin(\phi) & sin(\theta)cos(\phi) & 0 \\ cos(\theta)sin(\phi) & cos(\phi) & sin(\theta)sin(\phi) & 0 \\ -sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{105}$$

We need instead to initialize the register from 0 to 3 with the flattened version of the matrix

$$\begin{bmatrix} \cos(\theta)cos(\phi) & -sin(\theta) & cos(\theta)sin(\phi) & 0 \\ sin(\theta)cos(\phi) & cos(\theta) & sin(\theta)sin(\phi) & 0 \\ -sin(\phi) & 0 & cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{106}$$

which happens to correspond to the substitution $\phi \leftrightarrow \theta$.
In order to simulate the contraction we initialized the qubits in the desired states, applied the circuit described above and then measured every qubit. We ran 100k shots of the simulator and looked at the distribution of the output values, which provide an estimation of the probability associated to each state. In order to reduce the noise we repeated this whole procedure three times and took the average of the results. Here we show the result of the simulation, in particular the 3D plots describe the probability distributions associated to the four possible measurement outcomes $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, which are associated to the un-contracted qubits indexed by k and l) in our final state $|\Psi_{00}\rangle$, as functions of the rotation angles θ and ϕ:

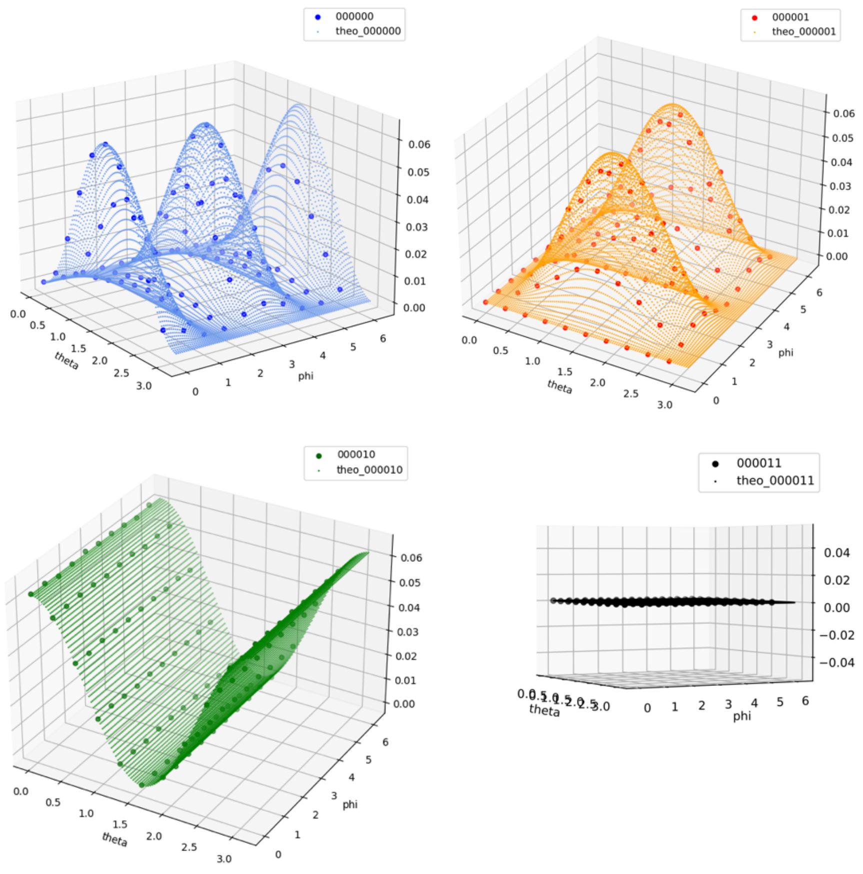


Figure 70: Results of the simulation of the Contractor Circuit in N=4 dimensions applied to **v**=[0,0,1,0] and the rotation matrix R 105. We can see that values of the estimated probabilities for component the final state $|\Psi_{00}\rangle$ (100k shots of the simulator, averaged over 5 attempts) as a function of the rotation angles of the matrix (θ, ϕ) behave as the squared components of the correctly rotated vector $R\mathbf{v} = [sin(\theta)cos(\phi), sin(\theta)sin(\phi), cos(\theta), 0]$.

The result seems in good agreement with the theoretical predictions, which according to equation (78) are, respectively, $(sin(\theta)cos(\phi))^2, (sin(\theta)sin(\phi))^2, cos(\theta)^2$ and 0.

8.4.5 The optimal number of dimensions

Every time we are working with M dimensional arrays we need to find the smallest integer n such that $2^n = N > M$ and set entries corresponding to the unused dimensions to zero. It is straightforward to notice that this algorithm is optimized when the dimension of the embedding for the word vectors is a power of two.

8.4.6 The Loss of Amplitude

In this 3D case, which is indeed a four dimensional one where we leave one dimension untouched, the probability of measuring our desired final state with the right amplitudes $|\Psi_{00}\rangle$ is $\frac{1}{4}$. More generally, we can consider a matrix to vector contraction in n dimension, which means flattening our indices in order to resize our tensors to a (NxN) matrix and a N vector that are encoded using respectively $2log_2(N)$ and $log_2(N) = n$-qubit registers. We denote with $a_i, i \in [1, n]$ the free indices we are not contracting (which after we flatten them become a single index with a range which is the sum of their ranges), while we call the indices we are contracting and summing over $b_i, i \in [1, n]$. The desired final state can be written as

$$|\Psi_{00}\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{a_1..a_n} [(\sum_{b_1..b_n} R_{a_1..a_n,b_1..b_n} M_{b_1..b_n})|a\rangle^{\otimes_i}|0^N\rangle] \tag{107}$$

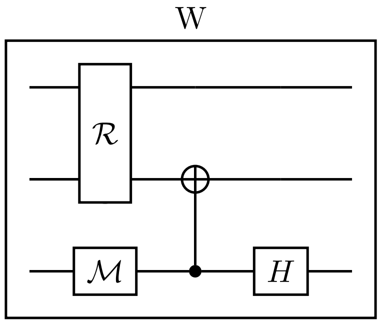
Whose probability of being measure is given by

$$P_{00} = \frac{1}{2^n} \sum_{a_1..a_n} (\sum_{b_1..b_n} R_{a_1..a_n,b_1..b_n} M_{b_1..b_n})^2 = \frac{1}{2^n} \sum_{a_1..a_n} (M'_{a_1..a_n})^2 = \frac{1}{2^n} \tag{108}$$

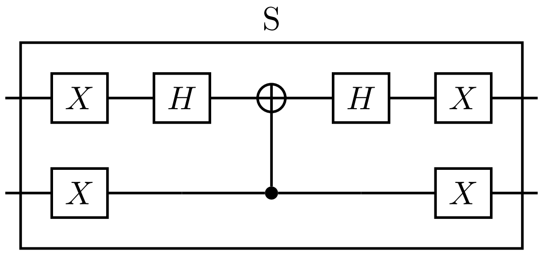
where we used the fact that our entries are real and that being M an unit vector and R a rotation matrix, which preserves the norm, $\mathrm{M}' = R\mathrm{M}$ is still a unit vector.

8.4.7 How to contrast the loss of amplitude

How we can see from previous section, the desired state $|\Psi_{00}\rangle$, where the contraction occurs, has an amplitude that (using isometries such as rotations) decreases at each step by a factor $2^{-\frac{n}{2}}$. The probability of measuring it scales hence with 2^{-n}, which could rapidly make the number of measurement required at the end of the circuit extremely large. Fortunately, we could use the amplitude amplification protocol in [Gro98] as a possible solution to the problem. Defining the composition of the encoding and contraction circuits as $\mathcal{W}$



(a) W circuit as a composition of the encoding and contractor circuits for N=2.



(b) S circuit in the N=2 case.

we repeatedly apply the circuit:

$$\mathcal{Q} = \mathcal{W}\mathcal{S}_0\mathcal{W}^\dagger\mathcal{S}. \tag{109}$$

Here $\mathcal{S}$ flips the sign of the states we wish to amplify, in our case the ones with the registers corresponding to the contracted indices in state $|0\rangle$, whereas $\mathcal{S}_0$ flips the sign of state $|0\rangle^{\otimes n \cdot N_{wires}}$. An explicit expression for S in the case of circuit 67 is given by 71b.
This method is a generalization of Grover's algorithm, to which it reduces when we choose $\mathcal{W}$ to be $H^{\otimes n}$.

After m iterations the state of the system is

$$Q^n |\Psi\rangle = \frac{sin[(2m+1)\theta)]}{\sqrt{P_{00}}} |\Psi_{00}\rangle + \frac{cos[(2m+1)\theta)]}{\sqrt{1-P_{00}}} |\Phi\rangle \tag{110}$$

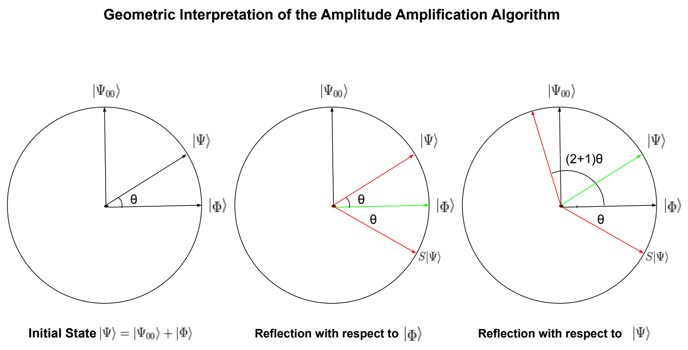
so the probability of measuring the desired state $|\Psi_{00}\rangle$ will be

$$\begin{cases} P_{00} = sin^2(\theta) \\ P_{00}^{new} = sin^2((2m+1)\theta)) \end{cases} \tag{111}$$

The ideal number of iterations is therefore $m = \frac{\pi}{4\theta}$ such that a good state (a basis state in the superposition $|\Psi_{00}\rangle$) is measured with a probability of at least $max(1 - P_{00}, P_{00})$.
In our case for large n we have

$$\begin{cases} P_{00} = 2^{-n} \\ \theta = arcsin(2^{-\frac{n}{2}}) \approx 2^{-\frac{n}{2}} \\ m \approx \pi 2^{\frac{n}{2}-2} \end{cases} \tag{112}$$

A visual interpretation of this algorithm is similar to the one used for Grover's, consisting in a series of pairs of reflections of the state of the system, one with respect to the unwanted or bad state $|\Phi\rangle$ (orthogonal to $|\Psi_{00}\rangle$), which is performed by $S = (2|\Phi\rangle\langle\Phi| - I)$, and the other one with respect to the initial state $\mathbf{\Psi}$ performed by $\mathcal{W}\mathcal{S}_{\prime}\mathcal{W}^\dagger = (2|\Psi\rangle\langle\Psi| - I)$.



Geometric Interpretation of the Amplitude Amplification Algorithm
$|\Psi_{00}\rangle$
$|\Psi\rangle$
$|\Phi\rangle$
θ
$S|\Psi\rangle$
(2+1)θ
Initial State $|\Psi\rangle = |\Psi_{00}\rangle + |\Phi\rangle$
Reflection with respect to $|\Phi\rangle$
Reflection with respect to $|\Psi\rangle$

Figure 72: Geometrical interpretation of the amplitude amplification algorithm

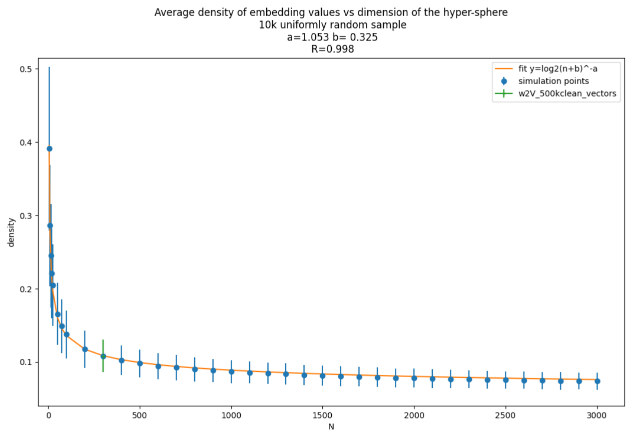


Figure 73: This graph shows our simulation point fit for the data density behavior as a function of n, where n-1 is the size of the hyper-spherical surface over which we uniformly sample the vectors. We can see how our data set, although it doesn't represent a uniform distribution, perfectly sits on the curve.

8.4.8 Cost Analysis of the Amplitude Amplification

It is important to notice that $2^n = N$ is the dimension of the embedding, which is fixed given the model. The number of required runs of the amplitude amplification algorithm after every contraction scales as $2^{\frac{n}{2}} = \sqrt{N}$. The amplitude amplification algorithm requires to apply the encoding gate and its adjoint at each step, which requires roughly $2 * \frac{log_2(n)}{\rho\epsilon}$ per step (as we are applying W and its adjoint), where ϵ is the desired average encoding error and ρ is the density of the embedding entries we previously defined in equation (67) 8.4.1.
Since this last parameter depends on the specific set of word embeddings, we tried to investigate its value on a set of random points on the n-sphere. We picked a 10k unit randomly distributed vectors and calculated their average density.

As we tried to fit this points with the inverse of a logarithm, more precisely with the function $y = log_2(N+0.325)^{1.053}$, the results seemed in good agreement (R=0.998). The last step was to calculate the density of our specific set of word embeddings, which appears to follow the same trend. From this plot we have and indication that the value of ρ on a set of unit vectors in N-dimensions is proportional to $\frac{1}{log_2(N)}$.
This means that the amount of not parallel steps $T_{encode}(N, \epsilon)$ we need to encode a vector embedding of length N scales as

$$T_{encode}(N, \epsilon) = \frac{log_2(n)}{\rho\epsilon} \propto \frac{log_2(log_2(N))log_2(N)}{\epsilon} \tag{113}$$

The final amount of not parall operation needed in order to perform the amplitude amplification after every contraction is:

$$C_{Amp}(N, \epsilon) = 2 \cdot T_{encode}(N, \epsilon) \cdot \sqrt{N} \propto \frac{log_2(log_2(N))log_2(N)}{\epsilon}\sqrt{N} \tag{114}$$

Although it is not trivial to determine how many alternative derivations a large period could have, if we consider that any grammatical ambiguity of type "ROL" represents a node in a binary tree of possibilities and that word level ambigu-

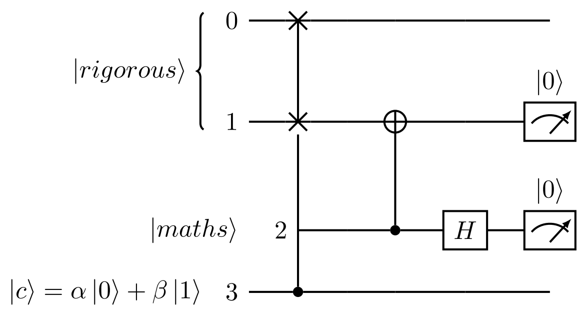


Figure 74: Circuit that performs a contraction of tensor indices with N=2 using a C-Swap

ities can in principle lead to multiple options, the amount of a priori possible interpretations to explore grows really fast.

One of the main disadvantages about this circuit is that the amplitude amplification requires to repeatedly prepare the initial state. This feature, due to the presence of the encoder gate E in $\mathcal{W}$, makes it really difficult to efficiently combine several copies of the circuit, since any predictive application should work without providing an encoding gate for every intermediate state of the algorithm. Composition is therefore possible only at the cost of removing the amplification step, reducing the amplitude by a factor $2^{\frac{n}{2}}$ at each contraction.

8.4.9 Simultaneous readings

The authors introduce then a way to simultaneously perform different contractions by means of an ancilla qubit that controls a SWAP gate. This circuit is analogous to the one we previously mentioned.

This circuit works through the action of the C-SWAP gate, but it is not always so trivial how to choose the permutations so that every desired contraction is performed. The main problem is that repeated applications of such a gate may make it difficult to take track of the right contraction order through the derivation. We therefore propose an alternative circuit which has the same effect but without using any C-SWAP gate.
We will make use of the circuit every time in the derivation we encounter an ambiguous string ”ROL”. As we explained, in those cases the algorithm has to choose between contracting the output wire in the middle either to the right or to the left. Here we show a sketch of the circuit:

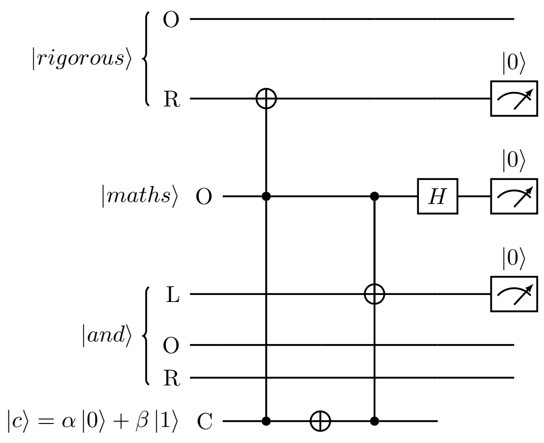
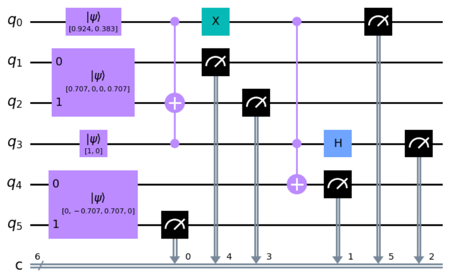


Figure 75: Conditional Contractor Circuit.

In order to simulate the action of this circuit we prepared three registers (2-1-2 qubits) initializing them respectively with states:

$$\begin{cases} |A\rangle = \frac{1}{\sqrt{(2)}}[cos(\theta)|00\rangle - sin(\theta)|01\rangle + sin(\theta)|10\rangle + cos(\theta)|11\rangle] \\ |B\rangle = \frac{1}{\sqrt{(2)}}[cos(\phi)|00\rangle + sin(\phi)|01\rangle - sin(\phi)|10\rangle + cos(\phi)|11\rangle] \\ |\Psi\rangle = |0\rangle \end{cases} \tag{115}$$

The circuit simulated on Qiskit is almost same circuit, with the control bit as first register and a tensor B with two indices instead of three:


q_0
q_1
q_2
q_3
q_4
q_5
c
6
$|\psi\rangle$ [0.924, 0.383]
$|\psi\rangle$ [0.707, 0, 0, 0.707]
$|\psi\rangle$ [1, 0]
$|\psi\rangle$ [0, −0.707, 0.707, 0]
X
H
0
4
3
1
5
2

In the simulation θ ranges in the interval $[0, \pi]$ while ϕ is defined at each step as $\frac{\pi}{2} - \frac{\theta}{2}$.

The control bit is set on state:

$$|C\rangle = \alpha|0\rangle + \beta|1\rangle$$

(116)

The parameters $|\alpha|^2$ and $|\beta|^2$ give us the probability of respectively collapsing into the state where we contract the output wire indices with the ones of the L wire or with the ones of the R wire . In a general setting calculations show that:

$$|\Psi\rangle = \alpha|1\rangle\{(\sum_{ij} A_{ij}|i\rangle|j\rangle)[\frac{1}{2}\sum_m(\sum_k O_k B_{km})|0\rangle|0\rangle|m\rangle + |\Phi_1\rangle]\} + \beta|0\rangle\{[\frac{1}{2}(\sum_i((\sum_k A_{ik}O_k)|i\rangle|0\rangle|0\rangle) + |\Phi_0\rangle](\sum_{lm} B_{lm}\sum|l\rangle|m\rangle)\} \quad (117)$$

In our specific case $O_k = \delta_{k0}$ and the matrices A and B are the rotation matrices encoded in states $|A\rangle$ and $|B\rangle$ in (65).

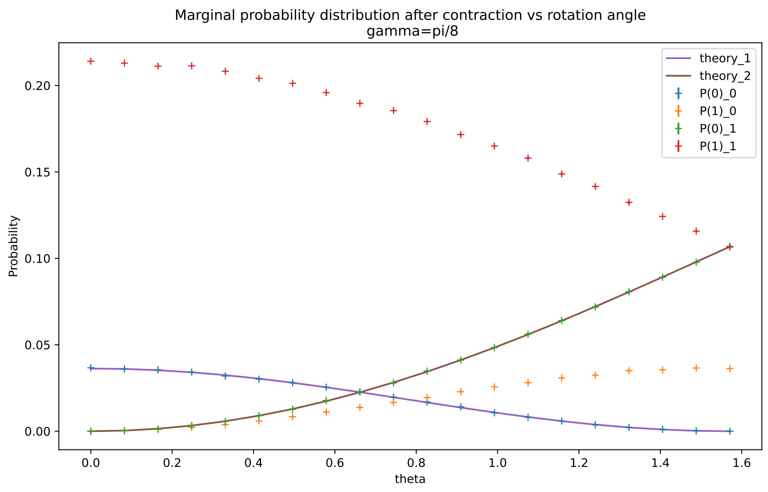


Figure 76: Plot of the marginal probability distribution defined below, the numbers indicate the measured registers

Here we plot the estimated marginal probabilities when $\alpha = cos(\gamma = \frac{\pi}{8})$ and $\phi(\theta) = \frac{\pi}{2} - \frac{\theta}{2}$:

$$\begin{cases} P(0)_0 = P(Std.Meas.Res_{0345} = 0000) \\ P(1)_0 = P(Std.Meas.Res_{0345} = 0100) \\ P(0)_1 = P(Std.Meas.Res_{0123} = 1000) \\ P(1)_1 = P(Std.Meas.Res_{0123} = 1100) \end{cases}$$

(118)

These results seem in good agreement with our theoretical predictions.

8.4.10 Comparing states

This circuit gives us a way to simultaneously perform alternative contractions without interchanging the position of the states. We can also weight these mutually exclusive choices by tuning the two amplitudes α and β. We would now like to have a reasonable way to choose these parameters: for example, analogously to what we did in the classical case, whenever we choose to contract a noun with an adjective or a conjunction, we could make the amplitudes of the two choices proportional to the likelihood of the associated words to appear together. This likelihood could be estimated by taking the inner product of the relative states at that moment of the contraction. We could start with a set of ancilla qubits prepared in the right superpositions of states, but that would require to know the outcomes of every contraction the circuit will perform in advance, completely defeating the purpose of the latter. We therefore started looking for a circuit that, given three states as input $\psi_1\ \phi\ \psi_2$, provided us with a qubit with amplitudes on states $|0\rangle$ and $|1\rangle$ that are proportional to $\langle\psi_1|\phi\rangle$ and$\langle\psi_2|\phi\rangle$.
One of the best known methods to compare states is the SWAP test, which was introduced first in the article [Bar+97] and that was then and later rediscovered by Harry Buhrman et al [Buh+01]. The circuit is the following:

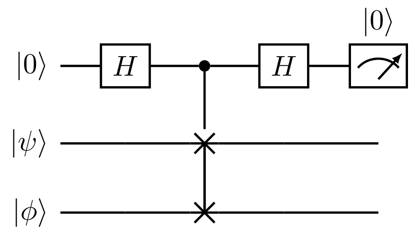


Figure 77: SWAP test circuit.

Right before the measurement ancilla qubit is in state:

$$|a\rangle = |0\rangle \frac{(|\psi\rangle\,|\phi\rangle + |\phi\rangle\,|\psi\rangle)}{2} + |1\rangle \frac{(|\psi\rangle\,|\phi\rangle - |\phi\rangle\,|\psi\rangle)}{2} \tag{119}$$

The probabilities of measuring 0 or 1 on the control qubit are respectively:

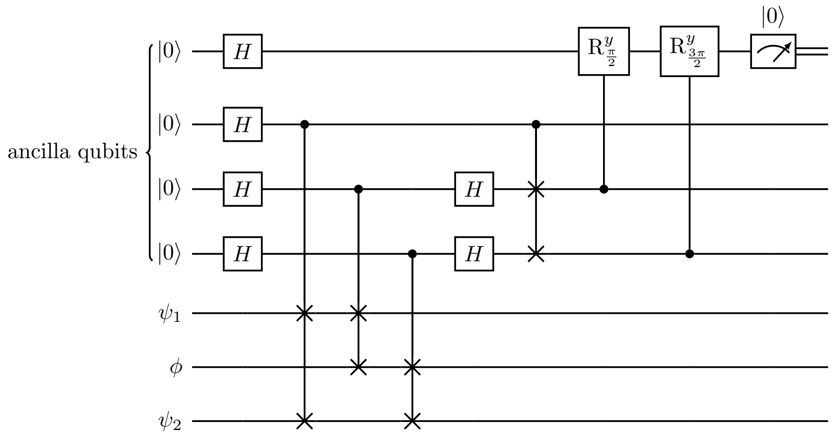

ancilla qubits
$|0\rangle$
H
$\mathrm{R}^y_{\frac{\pi}{2}}$
$\mathrm{R}^y_{\frac{3\pi}{2}}$
ψ_1
ϕ
ψ_2

Figure 78: Scheme of our comparison circuit.

$$\begin{cases} P(0) = \frac{1}{2} + \frac{\|\langle\psi|\phi\rangle\|^2}{2} \\ P(1) = \frac{1}{2} - \frac{\|\langle\psi|\phi\rangle\|^2}{2} \end{cases} \tag{120}$$

However, this test has both the disadvantage of not being symmetric in its action under the exchange of 0 and 1 and also of performing a comparison between only two states.
Anyways, after some educated guessing we found a more complex circuit that takes three states as input and outputs a state with the desired amplitudes.

Calculations show that the final right before the measurement should be:

$$\begin{aligned} |\Theta_0\rangle(1,2) = [&|00\rangle(|\psi_1\rangle|\phi\rangle|\psi_2\rangle + |\psi_1\rangle|\psi_2\rangle|\phi\rangle + |\phi\rangle|\psi_1\rangle|\psi_2\rangle + |\phi\rangle|\psi_2\rangle|\psi_1\rangle) + \\ &|11\rangle(|\psi_1\rangle|\phi\rangle|\psi_2\rangle - |\psi_1\rangle|\psi_2\rangle|\phi\rangle - |\phi\rangle|\psi_1\rangle|\psi_2\rangle + |\phi\rangle|\psi_2\rangle|\psi_1\rangle)] \end{aligned} \tag{121}$$

$$\begin{aligned} |\Psi\rangle = \frac{|0\rangle}{8}\{&|0\rangle[|\Theta_0\rangle(1,2) + \sqrt{2}|01\rangle(|\psi_1\rangle|\phi\rangle|\psi_2\rangle - |\psi_1\rangle|\psi_2\rangle|\phi\rangle + |\phi\rangle|\psi_1\rangle|\psi_2\rangle - |\phi\rangle|\psi_2\rangle|\psi_1\rangle)] + \\ &|1\rangle[|\Theta_0\rangle(2,1) + \sqrt{2}|01\rangle(|\psi_1\rangle|\phi\rangle|\psi_2\rangle - |\psi_1\rangle|\psi_2\rangle|\phi\rangle - |\phi\rangle|\psi_1\rangle|\psi_2\rangle + |\phi\rangle|\psi_2\rangle|\psi_1\rangle)]\} + \\ \frac{|1\rangle}{8}\{&|0\rangle[|\Theta_0\rangle(1,2) + \sqrt{2}|10\rangle(|\psi_1\rangle|\phi\rangle|\psi_2\rangle + |\psi_1\rangle|\psi_2\rangle|\phi\rangle - |\phi\rangle|\psi_1\rangle|\psi_2\rangle - |\phi\rangle|\psi_2\rangle|\psi_1\rangle)] + \\ &|1\rangle[|\Theta_0\rangle(2,1) + \sqrt{2}|10\rangle(|\psi_1\rangle|\phi\rangle|\psi_2\rangle - |\psi_1\rangle|\psi_2\rangle|\phi\rangle + |\phi\rangle|\psi_1\rangle|\psi_2\rangle - |\phi\rangle|\psi_2\rangle|\psi_1\rangle)]\} \end{aligned} \tag{122}$$

The probabilities of measuring 0 and 1 on the first qubit are hence :

$$\begin{cases} P(0) = \frac{1}{2} + \frac{3}{16}[\| \langle\psi_1|\phi\rangle \|^2 - \| \langle\psi_2|\phi\rangle \|^2] \\ P(1) = \frac{1}{2} - \frac{3}{16}[\| \langle\psi_1|\phi\rangle \|^2 - \| \langle\psi_2|\phi\rangle \|^2] \end{cases} \tag{123}$$

This circuit is therefore anti-symmetric under the exchange $\psi_1 \longleftrightarrow \psi_2$ but symmetric under the exchange $\psi_1 \longleftrightarrow \psi_2, |0\rangle \longleftrightarrow |1\rangle$.
Since the second term can be at most $\frac{3}{16}$, the most extreme cases give us probabilities $0.6875, 0.3125$ and $0.3125, 0.6875$. Using this circuit as a basic component we could deal with any occurrence of ambiguous strings "ROL" simultaneously by comparing the three closest states involved in the contraction and using the result as the control in our conditional contraction circuit.
Here we plot the results of the Qiskit simulation of the circuit. This time we choose the initial states to be:

$$\begin{cases} |\psi_1\rangle = |0\rangle \\ |\psi_2\rangle = |1\rangle \\ |\phi\rangle = cos(\theta)\,|0\rangle + sin(\theta)\,|1\rangle \end{cases} \tag{124}$$

So that:

$$\begin{cases} P(0) = \frac{1}{2} + \frac{3}{16}cos(2\theta) \\ P(1) = \frac{1}{2} - \frac{3}{16}cos(2\theta) \end{cases} \tag{125}$$

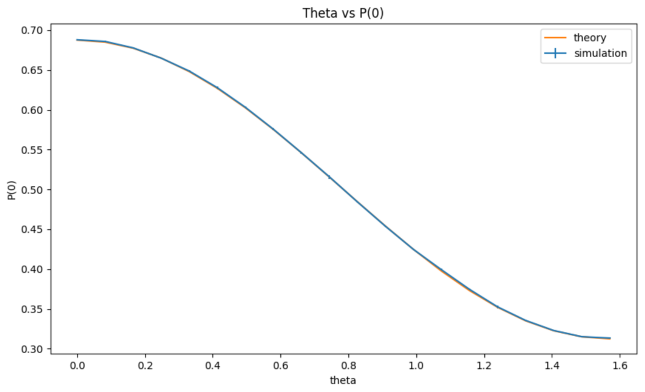
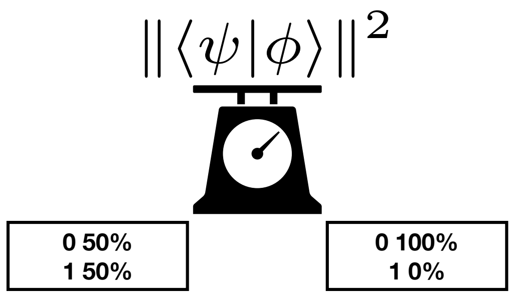
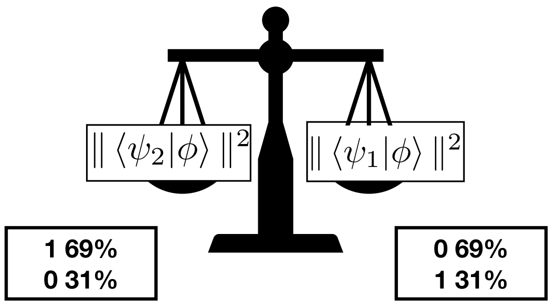


Figure 79: Result of the Qiskit simulation of our circuit for comparison, here we show the behaviour of P(0) as the the angle of initial state of ϕ ranges from 0 to $\frac{\pi}{2}$. We ran the simulator 100k times and average the results over 5 attempts. The curve is compared with the expected behaviour $y = \frac{1}{2} + \frac{3}{16}cos(2\theta)$ and seems in perfect agreement.



(a) Schematic illustration of the result of a swap test



$\| \langle\psi_2|\phi\rangle \|^2$
$\| \langle\psi_1|\phi\rangle \|^2$
1 69%
0 31%
0 69%
1 31%

(b) Schematic illustration of the result of our comparison circuit

8.4.11 Contractions between vectors and rotations

If we analyze the behaviour of the circuit above we notice that it compares states encoded into registers with the same amount of qubits. Yet, when we perform contractions going through a derivation, it is always between an output wire, which is a vector encoded in a n-register, and either a left or right input wire, which can be part of a state with 2n or 3n register. In the case of a (2n,n,2n) contraction between states:

$$\begin{cases} |\psi_1\rangle = \sum\limits_{i,j} R_{ij} |i\rangle |j\rangle \\ |\phi\rangle = \sum\limits_{k} O_k |k\rangle \\ |\psi_2\rangle = \sum\limits_{l,m} L_{lm} |l\rangle |m\rangle \end{cases} \tag{126}$$

we can show that the probability of measuring 0 or 1 on the control qubit are:

$$\begin{cases} P(0) = \frac{1}{2} + \frac{3}{16}[\sum\limits_{i}(\sum\limits_{k} R_{ik} O_k)^2 - \sum\limits_{m}(\sum\limits_{l} O_l L_{lm})^2] \\ P(1) = \frac{1}{2} - \frac{3}{16}[\sum\limits_{i}(\sum\limits_{k} R_{ik} O_k)^2 - \sum\limits_{m}(\sum\limits_{l} O_l L_{lm})^2] \end{cases} \tag{127}$$

This result appears problematic for our purpose because in our formalism both R and L are rotation matrices, which preserve the norm of vectors. This property makes the additional term in both the probabilities vanish.
A possible solution to this problem would be to provide every R or L register with an additional ”activity” register. This register could encode an array representing the activity distribution of the adjective associated to the R or L register.
Given an adjective A , a given corpus $\mathcal{V}$ and a set of word embeddings which we can clusterize in a set of centers $\{\mathbf{C_i}\}$, we can use the procedure described in the previous chapters to compute the activity distribution of A on the hyper sphere, which consists in a set of pairs vector-probability $\{(\mathbf{C_k}, p_{Ak})\}$ representing the cluster centers and their relative probability value defined as

$$\begin{cases} p_{Ai} = \frac{f_{Ai}}{\sum\limits_{k} f_{Ak}} = s^2_{Ai} \\ \sum\limits_{i}^{n_C} p_{Ai} = 1 \end{cases} \tag{128}$$

where f_{Ai} is the frequency with which the adjective is active on that specific region.
For example, if 30% of the times the bigram "tall" + noun occurs in the corpus we have that the word embedding corresponding to the noun belongs to a specific cluster region j on the n-sphere, with center $\mathbf{C_j}$ (by instance the "people" cluster) , we have $p_{Aj} = 0.3$ for A="tall".
From the the activity distribution we could prepare the state:

$$|\mathcal{D}_A\rangle = \sum_i s_{Ai} |C_i\rangle = \sum_i^{n_{clusters}} \sum_j^{N} s_{Ai} C_{ij} |j\rangle \tag{129}$$

This could be done by performing a contraction between the registers ($n_C = n_{clusters}$):

$$\begin{cases} |s_A\rangle = \sum_i^{n_C} s_{Ai} |i\rangle \\ |C\rangle = (\frac{1}{\sum_{lm} C_{lm}^2}) \sum_j^{n_C} \sum_k^{N} C_{jk} |j\rangle |k\rangle \end{cases} \tag{130}$$

The matrix C, which has the clusters arrays as rows, is the same for any adjective and can therefore be computed una tantum.

Another way to prepare the state could also be conditioning the action of different encoding gates using ancilla qubits, whose distribution should reflect our set of amplitudes $\{s_{Ai}\}$. The control qubit could be approximately prepared using the procedure we previously described (the encoder gate), but also exactly using the procedure described in the article Efficient Scheme for Initializing a Quantum Register with an Arbitrary Superposed State by Gui-Lu Long and Yang Sun
For example, in case of two active clusters with desired amplitudes s_1 and s_2 we have:

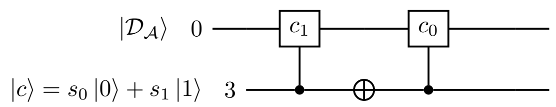

$|\mathcal{D}_\mathcal{A}\rangle$ 0
$|c\rangle = s_0 |0\rangle + s_1 |1\rangle$ 3
c_1
c_0

Figure 81: Conditional encoder with two alternatives.

whereas with three of options we can use :

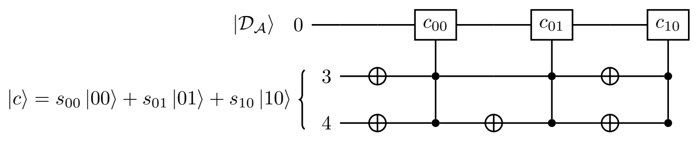

$|\mathcal{D}_\mathcal{A}\rangle$ 0
$|c\rangle = s_{00}|00\rangle + s_{01}|01\rangle + s_{10}|10\rangle$
3
4
c_{00}
c_{01}
c_{10}

Figure 82: Conditional encoder with three alternatives.

This "activity" registers represent the activity distribution of the adjective on the hyper-sphere and they therefore reflect its nature: the more vague an adjective is the flatter its distribution will be. In other words, whenever we use our Libra circuit to compare the output register with the two input ones, the algorithm statistically prefers the contraction with the adjective (or in general the transformation) whose activity distribution has a peak close to the position of the vector encoded by the output register.
A possible different approach is to take the word embedding associated to the adjective by the model, hence the one that is already present in our set of 500k pre-trained vectors, and use it as a comparison vector for contractions from the left and from the right with the adjective matrix. This procedure could be further improved by training density matrices instead of vectors, in order to capture directionality and therefore distinguish between left and right.

8.4.12 How to incorporate grammar rules

Now that we know how to take into account the similarity between an input and an output wire, we would also like to consider the grammatical correctness of the sentence. This can be statistically achieved by considering the wire types we mentioned in subsection 8.3.1. We can add a set of "type" qubits to each register, choosing them to be orthonormal to each other. In our case for example we could choose the map:

$$\begin{cases} NS \longrightarrow |00\rangle \\ N \longrightarrow |01\rangle \\ S \longrightarrow |10\rangle \\ ACT \longrightarrow |11\rangle \end{cases} \tag{131}$$

Since this vectors are orthonormal, whenever our Libra circuit compares an incompatible pair output-input, the fidelity will automatically be zero, assigning to that contraction a lower probability (in the most extreme scenario 0.31).
Going back to our example 63 ,the derivations "(old women)and(men retire)", "old((women and men)retire)" and "old(women and)(men retire)" would all have suppressed probabilities in the final superposition of states since they contract orthogonal type registers.

Wrapping all together, every R-L input and output register will have, respectively, the following components:
A vector sub-register $|A\rangle$, which encodes the entries of the word embedding, of a matrix or of a tensor depending on the particular case, a type sub-register $|T_A\rangle$ with values that belong, in our example, to the set $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ and correspond respectively to the types NS,N,S,ACT, and finally ,only in the case of a R or L Input register, an activity sub-register $|\mathcal{D}_A\rangle$ defined in equation 129. This last register could also encode just the word embedding of the given

adjective, adverb or conjunction, as we explained in the previous section 7.4.11.

$$\text{(R or L)-input}\begin{cases} |A\rangle \equiv \\ |T_A\rangle \equiv \\ |\mathcal{D}_\mathcal{A}\rangle \equiv \end{cases}$$

$$\text{output}\begin{cases} |O\rangle \equiv \\ |T_O\rangle \equiv \end{cases}$$

Figure 83: Scheme of the sub-registers of an input and an output wire.

8.4.13 Three index tensors : AND

The last kind of transformation, except for some special cases like double transitive verbs, has a L input, an output and a R input and corresponds to a three index tensor. In our example "old women and men retire", and belongs to this class of transformations since it can accept two NS registers, from the left and from the right, and output another NS register, which is supposed to be the vector equivalent of a logical end between the two concepts carried by the nouns. A possible interpretation of a logical end in terms of word embeddings could be the average of the two input vectors associated to the nouns. In our case this should provide us with a point on the sphere that lies in middle point between either the nouns "women" and "men" in one case, or "old women" and "men" in the other, depending on the particular derivation we consider. After some calculation we realized that it is impossible to have a tensor which doesn't depend on the input vectors and performs:

$$(\mathbf{v}A\mathbf{w})_k = \sum_i \sum_j v_i A_{ikj} w_j = \frac{v_k + w_k}{2} \tag{132}$$

A good candidate could be $\frac{\delta_{ik}+\delta kj}{2}$, which gives us:

$$(\mathbf{v}(\frac{\delta_{ik}+\delta kj}{2})\mathbf{w})_k = \sum_i \sum_j \frac{1}{2}(v_i \delta_{ik} w_j + v_i \delta_{kj} w_j) = \frac{1}{2}(v_k \sum_j w_j + w_k \sum_i v_i) \tag{133}$$

This result is similar to what we wanted, except for the terms $\sum_i v_i$ and $\sum_j w_j$, which doesn't equal to one. We could try to correct this terms using the cluster centers: as we tried to locally approximate the two index tensors (matrices) related to the adjective in a neighbourhood of each cluster center, we could generalize this idea to three index tensors by considering pairs of cluster regions, one as the left input and one as the right one. If we obtained $n_{cluster}$ cluster centers on the n-sphere for the nouns, we could build $n_{cluster} \cdot n_{cluster}$ different local representation of a three index tensor like "AND". We could then use the cluster center vectors to correct our expression in (126). If we want to apply "AND" to two nouns with word embeddings $\mathbf{v}$ and $\mathbf{w}$, with the first belonging to cluster i and the second one to cluster j, with cluster centers respectively $\mathbf{X_{CMi}}$ and $\mathbf{X_{CMj}}$, we could assign to the local representation of "AND"

$A_[ij]$ the following expression :

$$A_[ij]_{a,k,b} = \frac{1}{2}(\delta_{ak}X_{CMjb} + \delta kbX_{CMia}) \tag{134}$$

This way the contraction would give:

$$\frac{1}{2}\sum_a\sum_b[v_a(\delta_{ak}X_{CMjb} + \delta kbX_{CMia})w_b] = \frac{1}{2}(v_k\sum_b w_bX_{CMjb} + w_k\sum_a v_aX_{CMia}) \approx \frac{v_k + w_k}{2} \tag{135}$$

Since $\mathbf{v} \approx \mathbf{X_{CMi}}$ $\mathbf{w} \approx \mathbf{X_{CMj}}$ if we choose the clusters to be small enough and $\sum_a v_aX_{CMia} \approx \sum_a v_a^2 = 1$.

8.4.14 R-O-L contraction

According to what we explained in the previous sections, every time we encounter a ROL ambiguity in our derivation we can deal with it combining the comparison or ”Libra” circuit and the conditional contractor one. The comparison circuit takes as input the activity and type sub-registers of the input wires L and R as well as the vector and type sub-registers of the output wire O. As we showed before, the circuit also requires additional ancilla qubits (4 in the case of N=2). The output of the Libra circuit is a quantum state in the superposition 122

$$|\Psi\rangle = |0\rangle|\Psi_R\rangle + |1\rangle|\Psi_L\rangle \tag{136}$$

$$\tag{137}$$

where $|\Psi_R\rangle$ and $|\Psi_L\rangle$ are
:

$$\begin{aligned}|\Psi_R\rangle = \frac{1}{8}\{&|0\rangle[|\Theta_0\rangle(1,2) + \sqrt{2}|01\rangle(|\psi_1\rangle|\phi\rangle|\psi_2\rangle - |\psi_1\rangle|\psi_2\rangle|\phi\rangle + |\phi\rangle|\psi_1\rangle|\psi_2\rangle - |\phi\rangle|\psi_2\rangle|\psi_1\rangle)]+\\ &|1\rangle[|\Theta_0\rangle(2,1) + \sqrt{2}|01\rangle(|\psi_1\rangle|\phi\rangle|\psi_2\rangle - |\psi_1\rangle|\psi_2\rangle|\phi\rangle - |\phi\rangle|\psi_1\rangle|\psi_2\rangle + |\phi\rangle|\psi_2\rangle|\psi_1\rangle)]\}\end{aligned} \tag{138}$$

$$\begin{aligned}|\Psi_L\rangle = \frac{1}{8}\{&|0\rangle[|\Theta_0\rangle(1,2) + \sqrt{2}|10\rangle(|\psi_1\rangle|\phi\rangle|\psi_2\rangle + |\psi_1\rangle|\psi_2\rangle|\phi\rangle - |\phi\rangle|\psi_1\rangle|\psi_2\rangle - |\phi\rangle|\psi_2\rangle|\psi_1\rangle)]+\\ &|1\rangle[|\Theta_0\rangle(2,1) + \sqrt{2}|10\rangle(|\psi_1\rangle|\phi\rangle|\psi_2\rangle - |\psi_1\rangle|\psi_2\rangle|\phi\rangle + |\phi\rangle|\psi_1\rangle|\psi_2\rangle - |\phi\rangle|\psi_2\rangle|\psi_1\rangle)]\}\end{aligned} \tag{139}$$

where $|\psi_1\rangle = |\mathcal{D}_R, T_R\rangle$,$|\psi_2\rangle = |\mathcal{D}_L, T_L\rangle$ and $|\phi\rangle = |O, T_O\rangle$
The first qubit of $|\Psi\rangle$ has probability of being measured in the $|0\rangle$ or $|1\rangle$ state respectively given by

$$\begin{cases}P(0) = \frac{1}{2} + \frac{3}{16}[\| \langle\mathcal{D}_R, T_R|O, T_O\rangle \|^2 - \| \langle\mathcal{D}_L, T_L|O, T_O\rangle \|^2]\\ P(1) = \frac{1}{2} - \frac{3}{16}[\| \langle\mathcal{D}_R, T_R|O, T_O\rangle \|^2 - \| \langle\mathcal{D}_L, T_L|O, T_O\rangle \|^2]\end{cases} \tag{140}$$

.
Therefore $P(0)$ will be greater than $P(1)$ if the combination of activity and type sub-registers associated to tensor OR, which is contracting from the left, with O is more similar to $|O, T_O\rangle$ than the one corresponding to the tensor LO, which is contracting from the right and vice versa. We can use this first qubit as the control qubit of the conditional contractor between the vector and type sub-registers of the R and L wires and the same registers of the output wire O. In order to be coherent with our notation, we first need to apply a NOT gate to the first qubit of $|\Psi\rangle$ obtaining

$$|\Psi\rangle = |1\rangle |\Psi_R\rangle + |0\rangle |\Psi_L\rangle \tag{141}$$

(142)

The application of conditional contractor circuit therefore leaves us with the state

$$\begin{aligned} |\Psi_{final}\rangle = |0\rangle |\Psi_R\rangle \{[\frac{1}{2}(\sum_i((\sum_k(OR)_{ik}O_k)|i\rangle|0\rangle|0\rangle) + |\Phi_0\rangle](\sum_{lm}(LO)_{lm}\sum|l\rangle|m\rangle)\}+ \\ |1\rangle |\Psi_L\rangle \{(\sum_{ij}(OR)_{ij}|i\rangle|j\rangle)[\frac{1}{2}\sum_m(\sum_k O_k(LO)_{km})|0\rangle|0\rangle|m\rangle + |\Phi_1\rangle]\} \end{aligned} \tag{143}$$

This procedure has hence left us with a final state which has the two contractions in a superposition and whose weights are proportional to the likelihood of one alternative with respect to the other. Since the comparison is a destructive process, this process consumes two copies of the output register the ones (one for the comparison and the other for the contraction), a copy of both $|\mathcal{D}_R, T_R\rangle$ and $|\mathcal{D}_L, T_L\rangle$ (again for the comparison) as well as a copy of $|\mathcal{OR}, T_R\rangle$ and $|\mathcal{LO}, T_L\rangle$ for the contraction.

8.4.15 R-O and O-L contractions

For what concerns the contraction between an input and an output wire, which can happen between R and O or between O and L, there aren't any ambiguities involved. However, it would make sense to give the circuit a probability of performing (or not) the contraction at the given step which is proportional to the similarity between the two registers, as in the case of the R-O-L contraction. This could be achieve by using a slightly modified version of the swap test between the sub-registers of the L or R input wire ($|\mathcal{D}_L, T_L\rangle$ or $|\mathcal{D}_R, T_R\rangle$) and the ones of the output wire $|O, T_O\rangle$. If we consider for instance a O-L contraction, when $\langle\mathcal{D}_L, T_L|O, T_O\rangle = 0$ the normal Swap test gives for the ancilla qubit probabilities

$$\begin{cases} P(0) = P(1) = 0.5, \langle\mathcal{D}_L, T_L|O, T_O\rangle = 0 \\ P(0) = 1; P(1) = 0, \langle\mathcal{D}_L, T_L|O, T_O\rangle = 1 \end{cases} \tag{144}$$

Even when the states are orthogonal, for example because the contraction is grammatically forbidden, we still have a 50% chance of contracting the indices. We can improve this result by modifying the Swap test in the following manner
Changing the angle θ we can also change $P_0(\theta, A)$,$P_1(\theta, A)$, where we define

$$A = \langle\mathcal{D}_L, T_L|O, T_O\rangle \tag{145}$$

If we constraint the final state of the system including the additional ancilla qubit to be of the form

$$(cos(\theta)|0\rangle + sin(\theta)|1\rangle)|0\rangle|S\rangle + (cos(\phi)|0\rangle + sin(\phi)|1\rangle)|1\rangle|A\rangle \tag{146}$$

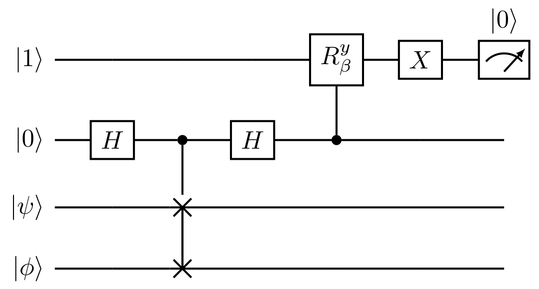


Figure 84: Modified version of the swap test to compare states.

with

$$\begin{cases} |S\rangle = \frac{|\psi\rangle|\phi\rangle+|\psi\rangle|\phi\rangle}{2} \\ |A\rangle = \frac{|\psi\rangle|\phi\rangle-|\psi\rangle|\phi\rangle}{2} \end{cases} \tag{147}$$

We have the following expressions for the probabilities P(0) and P(1) on the additional ancilla qubit:

$$\begin{cases} P_0(\theta,\phi,A) = cos^2(\theta)\frac{1+A}{2} + cos^2(\phi)\frac{1-A}{2} \\ P_1(\theta,\phi,A) = sin^2(\theta)\frac{1+A}{2} + sin^2(\phi)\frac{1-A}{2} \end{cases} \tag{148}$$

We would like to have $P_0(A=0)=1$ but this would imply $cos^2(\theta)=cos^2(\phi)=1$ and therefore $P_0(A)=1$ for every A, even when A=1, which is not a reasonable result. If we impose $P_0(1-A)=P_1(A)$ we get $cos^2(\phi)=2-3cos^2(\theta)$ with the additional constraint of having $-\sqrt{(\frac{2}{3})} \le cos(\theta) \le \sqrt{(\frac{2}{3})}$. The probabilities become

$$\begin{cases} P_0(\theta,A) = sin^2(\theta) + A(2cos^2(\theta)-1) \\ P_1(\theta,A) = cos^2(\theta) + A(2sin^2(\theta)-1) \end{cases} \tag{149}$$

The choice that maximizes $P_0(A=0)=P_1(A=1)=sin^2(\theta)$ while fulfilling the constraint $-\sqrt{(\frac{2}{3})} \le cos(\theta) \le \sqrt{(\frac{2}{3})}$ is

$$\cos(\theta) = \sqrt{(\frac{2}{3})}, \cos(\phi) = 0 \tag{150}$$

which gives the following probabilities for the ancilla qubit:

$$P_0(\theta,A) = 1/3 + \frac{A}{3} P_1(\theta,A) = 2/3 - \frac{A}{3} \tag{151}$$

In terms of the angle β in fig.84 this means choosing $cos(\theta)=cos(\frac{\pi}{2}-\beta)=sin(\beta)=\sqrt{\frac{2}{3}}$ and after applying the NOT gate we finally have

$$\begin{cases} P_0(\theta,A) = 2/3 - \frac{A}{3} \\ P_1(\theta,A) = 1/3 + \frac{A}{3} \end{cases} \tag{152}$$

which suppresses the probabilities of the derivations involving contractions between orthogonal states by a factor $\frac{1}{3}$.

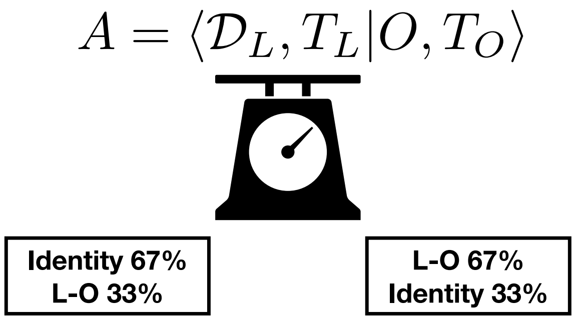


Figure 85: Schematic illustration of the output of our modified swap test in the case of a LO contraction.

9 Conclusions

The procedure described in chapter 7 to obtain a representation of the adjective can in principle be generalized to any type of transformation on the noun space, in the sense of the categorial grammars.
We were unable to train our representations on a large enough corpus of text due to limitations in computational and storage resources, despite the fact that we find the idea of testing this approach on a larger scale appealing. This is especially true for the visual representation of the activity distribution of the adjectives, because a large portion of them only had low frequencies on a very small number of clusters. Another future study could be on further specific applications of the local orthogonal approximation of the action of the adjectives. For what concerns the quantum section, as we mentioned, it is in principle possible to use the comparison and the conditional contractor circuits to simultaneously perform different derivations of a sentence. Due to the process of conditional contraction, each state in the final superposition would ideally have an amplitude that is proportional to the product of the single probabilities of the contractions, specific to the derivation it encodes. Grammatically incorrect derivations are suppressed in amplitude through the process by the orthogonality of their type sub-registers. However, in order to combine several copies of this circuit into a larger one, we need to deal with the problem of the loss of amplitude. As we showed in the dedicated section, it is possible to use amplitude amplification to contrast this phenomenon. Nevertheless, since this algorithm requires repeatedly preparing the original configuration of states that we want to modify (by amplifying the desired states), it is not practical to perform it as an intermediate step, as this would imply executing the same parts of the circuit several times. It is therefore important to investigate how to efficiently incorporate this algorithm for circuits of a significant size. Another idea for future improvements would be the usage of control qubits in order to avoid the double contraction of indices, which seems less challenging than the previous task. Finally, in order to combine several pairs of the comparison and contraction circuits, since every comparison step consumes a copy of each involved state, a possible strategy is to run the circuit on multiple copies of the initial registers. At every comparison-contraction step, one of the copies is consumed while the rest are contracted and go to the next step in the circuit. As we mentioned when discussing the possible alternatives to our tensor representation, we once again notice that our approach is modular, and it is therefore possible to experiment with different methods to obtain the representation that we will then use as the classical input for the quantum approach. All the challenges we have mentioned above are not trivial, but hopefully not impossible to overcome.

Bibliography

[Har54] Zellig S. Harris. "Distributional Structure". In: *¡i¿WORD¡/i¿* 10.2-3 (1954), pp. 146–162. DOI: 10.1080/00437956.1954.11659520. eprint: https://doi.org/10.1080/00437956.1954.11659520. URL: https://doi.org/10.1080/00437956.1954.11659520.

[58] In: (1958). DOI: https://doi.org/10.1037/h0042519. URL: https://doi.org/10.1016%2Fj.neunet.2014.09.003.

[Blu60] L. E. Blumenson. ""A Derivation of N-Dimensional Spherical Coordinates."". In: *The American Mathematical Monthly, vol. 67, no. 1* (1960), pp. 63–66. DOI: https://doi.org/10.2307/2308932.

[Pag84] Thomas J Page Jr. "Multivariate statistics: A vector space approach". In: *JMR, Journal of Marketing Research (pre-1986)* 21.000002 (1984), p. 236.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

[Elm90] Jeffrey L. Elman. "Finding structure in time". In: *Cognitive Science* 14.2 (1990), pp. 179–211. ISSN: 0364-0213. DOI: https://doi.org/10.1016/0364-0213(90)90002-E. URL: https://www.sciencedirect.com/science/article/pii/036402139090002E.

[Ger91] Kneller Gerald R. "Superposition of Molecular Structures using Quaternions". In: *Molecular Simulation* 7.1-2 (1991), pp. 113–119. DOI: 10.1080/08927029108022453.

[Pel94] Francis Jeffry Pelletier. "The Principle of Semantic Compositionality". In: *Topoi* 13.1 (Mar. 1994), pp. 11–24. ISSN: 1572-8749. DOI: 10.1007/BF00763644. URL: https://doi.org/10.1007/BF00763644.

[Est+96] Martin Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[Bar+97] Adriano Barenco et al. "Stabilization of Quantum Computations by Symmetrization". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1541–1557. DOI: 10.1137/S0097539796302452. eprint: https://doi.org/10.1137/S0097539796302452. URL: https://doi.org/10.1137/S0097539796302452.

[Gro98] Lov K. Grover. "Quantum Computers Can Search Rapidly by Using Almost Any Transformation". In: *Physical Review Letters* 80.19 (May 1998), pp. 4329–4332. DOI: 10.1103/physrevlett.80.4329. URL: https://doi.org/10.1103%2Fphysrevlett.80.4329.

[Loa98] Ralph Loader. *Notes on Simply Typed Lambda Calculus*. 1998.

[JM00] Daniel Jurafsky and James H Martin. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. 2000.

[Buh+01] Harry Buhrman et al. "Quantum Fingerprinting". In: *Physical Review Letters* 87.16 (Sept. 2001). DOI: 10.1103/physrevlett.87.167902. URL: https://doi.org/10.1103%2Fphysrevlett.87.167902.

[AJL05] Dorit Aharonov, Vaughan Jones, and Zeph Landau. "A Polynomial Quantum Algorithm for Approximating the Jones Polynomial". In: (2005). DOI: 10.48550/ARXIV.QUANT-PH/0511096. URL: https://arxiv.org/abs/quant-ph/0511096.

[DX08] H. Quynh Dinh and Liefei Xu. “Measuring the Similarity of Vector Fields Using Global Distributions”. In: *Structural, Syntactic, and Statistical Pattern Recognition*. Ed. by Niels da Vitoria Lobo et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 187–196. ISBN: 978-3-540-89689-0.

[BZ10] Marco Baroni and Roberto Zamparelli. “Nouns are Vectors, Adjectives are Matrices: Representing Adjective-Noun Constructions in Semantic Space”. In: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2010, pp. 1183–1193. URL: https://aclanthology.org/D10-1115/.

[TMV12] Janssen T.M.V. *Compositionality: its historic context*. Oxford: Oxford University Press, 2012. ISBN: 9780199541072.

[Mik+13a] Tomas Mikolov et al. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. DOI: 10.48550/ARXIV.1310.4546. URL: https://arxiv.org/abs/1310.4546.

[Mik+13b] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. DOI: 10.48550/ARXIV.1301.3781. URL: https://arxiv.org/abs/1301.3781.

[Nic13] Bogdan Nica. “The Mazur-Ulam theorem”. In: (2013). DOI: 10.48550/ARXIV.1306.2380. URL: https://arxiv.org/abs/1306.2380.

[HRK14] Felix Hill, Roi Reichart, and Anna Korhonen. *SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation*. 2014. DOI: 10.48550/ARXIV.1408.3456. URL: https://arxiv.org/abs/1408.3456.

[SW15] Adriaan M. J. Schakel and Benjamin J. Wilson. *Measuring Word Significance using Distributed Representations of Words*. 2015. URL: https://arxiv.org/abs/1508.02297.

[Sch15] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003. URL: https://doi.org/10.1016%2Fj.neunet.2014.09.003.

[Sta15] R.P. Stanley. *Catalan Numbers*. Cambridge University Press, 2015. ISBN: 9781316299586. URL: https://books.google.nl/books?id=i5QSBwAAQBAJ.

[BBP16] Saroj Biswas, Monali Bordoloi, and Biswajit Purkayastha. “Review on Feature Selection and Classification using Neuro-Fuzzy Approaches”. In: *International Journal of Applied Evolutionary Computation* 7 (Oct. 2016), pp. 28–44. DOI: 10.4018/IJAEC.2016100102.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. “6.5 Back-Propagation and Other Differentiation Algorithms.” In: MIT Press, 2016, pp. 200–220. ISBN: 9780262035613.

[Rud16] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016. DOI: 10.48550/ARXIV.1609.04747. URL: https://arxiv.org/abs/1609.04747.

[Vas+17] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.

[Dow+18] Meghan Dowling et al. “SMT versus NMT: Preliminary comparisons for Irish”. In: *Proceedings of the AMTA 2018 Workshop on Technologies for MT of Low Resource Languages (LoResMT 2018)*. Boston, MA: Association for Machine Translation in the Americas, Mar. 2018, pp. 12–20. URL: https://aclanthology.org/W18-2202.

[JP18] Mu Jiaqi and Viswanath Pramod. “ALL-BUT-THE-TOP: SIMPLE AND EFFECTIVE POST- PROCESSING FOR WORD REPRESENTATIONS”. In: (2018).

[Kho+18] Mikhail Khodak et al. *A La Carte Embedding: Cheap but Effective Induction of Semantic Feature Vectors*. 2018. DOI: 10.48550/ARXIV.1805.05388. URL: https://arxiv.org/abs/1805.05388.

[San+19] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *ArXiv* abs/1910.01108 (2019).

[SW19] Oded Schwartz and Elad Weiss. "Revisiting a Computation of Matrix Chain Products". In: *SIAM Journal on Computing* 48.5 (2019), pp. 1481–1486. DOI: 10.1137/18M1195401. eprint: https://doi.org/10.1137/18M1195401. URL: https://doi.org/10.1137/18M1195401.

[Bro+20] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. DOI: 10.48550/ARXIV.2005.14165. URL: https://arxiv.org/abs/2005.14165.

[WSC20] Gijs Wijnholds, Mehrnoosh Sadrzadeh, and Stephen Clark. "Representation Learning for Type-Driven Composition". In: *Proceedings of the 24th Conference on Computational Natural Language Learning, CoNLL 2020, Online, November 19-20, 2020*. Ed. by Raquel Fernández and Tal Linzen. Association for Computational Linguistics, 2020, pp. 313–324. DOI: 10.18653/v1/2020.conll-1.24. URL: https://doi.org/10.18653/v1/2020.conll-1.24.

[CMS22] A. D. Correia, M. Moortgat, and H. T. C. Stoof. "Quantum computations for disambiguation and question answering". In: *Quantum Information Processing* 21.4 (Mar. 2022), p. 126. ISSN: 1573-1332. DOI: 10.1007/s11128-022-03441-9. URL: https://doi.org/10.1007/s11128-022-03441-9.

[Com+] Phillip Compeau et al. "chapter 1: ANALYZING THE CORONAVIRUS SPIKE PROTEIN". In: *Biological Modeling: A Free Online Course*. Carnagie Mellon University. URL: https://biologicalmodeling.org/coronavirus/accuracy.

[Jay] Alammar Jay. *Visualizing machine learning one concept at a time*. URL: https://jalammar.github.io/illustrated-transformer/.

[Kit] Kit Kittelstad. *Examples of Ambiguity in Language and Literature*. URL: https://examples.yourdictionary.com/reference/examples/examples-of-ambiguity.html.

[ADC22] A.D.Correia. *Quantum Distributional Semantics, Quantum Algorithms Applied to Natural Language Processing*. Utrecht University: ProeftshriftMaken, September 2022. DOI: doi.org/10.33540/631.

[AG69] Allen and Beatrice Gardner. "Teaching Sign Language to a Chimpanzee". In: *Science* (August 1969). DOI: 10.1126/science.165.3894.664.

[Chr17] McCormick Chris. *Word2Vec Tutorial - The Skip-Gram Model*. 11 Jan 2017. URL: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/.

[DJ23] Jurafsky Daniel and H. Martin James. *Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 7 January 2023. URL: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf.

[Hsi17] Wu-Yi Hsiang. *Lectures On Lie Groups (Second Edition) (Series on University Mathematics, Band 9)*. World Scientific, April 07, 2017. URL: http://infinity.wecabrio.com/9814740713-lectures-on-lie-groups-second-edition-series-on-u.pdf.

[Joa58] Lambek Joachim. "The Mathematics of Sentence Structure". In: *The American Mathematical Monthly* (March 1958), pp. 154–170. URL: http://links.jstor.org/sici?sici=0002-9890%28195803%2965%3A3%3C154%3ATMOSS%3E2.0.CO%3B2-7.

[Kab76] Wolfgang Kabsch. "A solution for the best rotation to relate two sets of vectors." In: (12 April 1976).

[Sah22] Ashhab Sahel. “Quantum state preparation protocol for encoding classical data into the amplitudes of a quantum information processing register’s wave function”. In: *PHYSICAL REVIEW RESEARCH* (3 February 2022). DOI: `10.1103/PhysRevResearch.4.013091`.

[Yas15] Sudo Yasutada. *Lecture Notes on Advanced Semantic Theory B*. University College London, Spring 2015. URL: `https://www.ucl.ac.uk/~ucjtudo/teaching/AST-B.pdf`.